DOULOS

Delivering KnowHow www.doulos.com

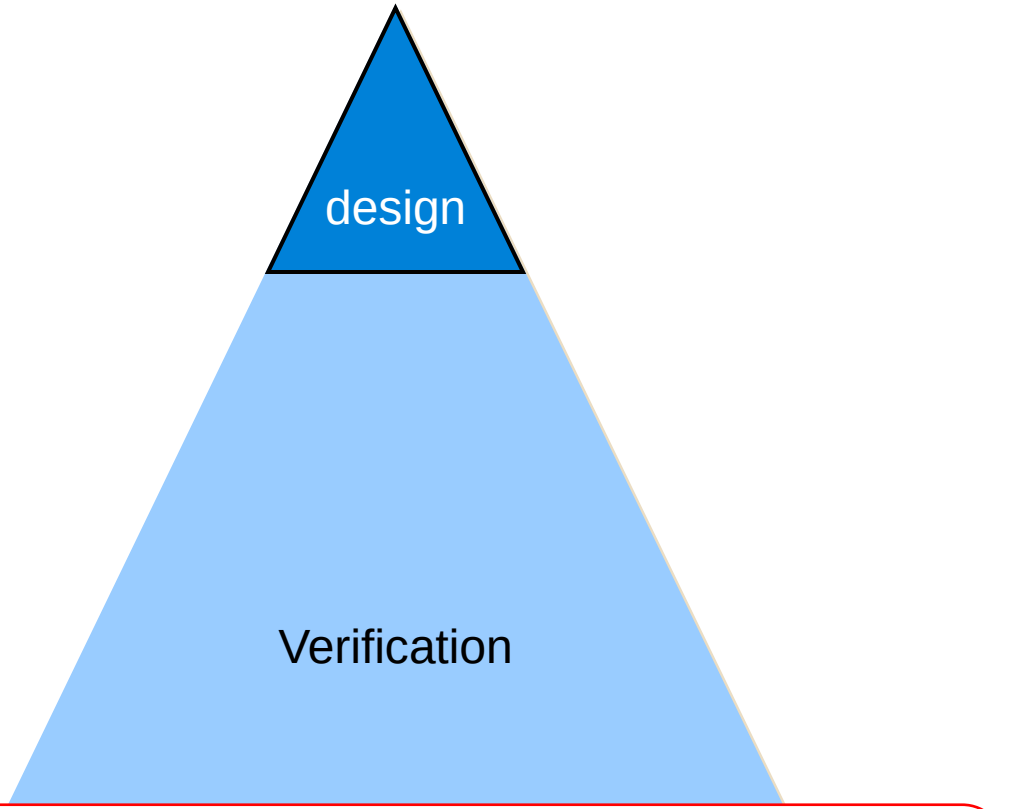
Navigating the Verification Maze

Presenter: Matthew Taylor
Senior Member of Technical Staff

Copy

Functional Verification

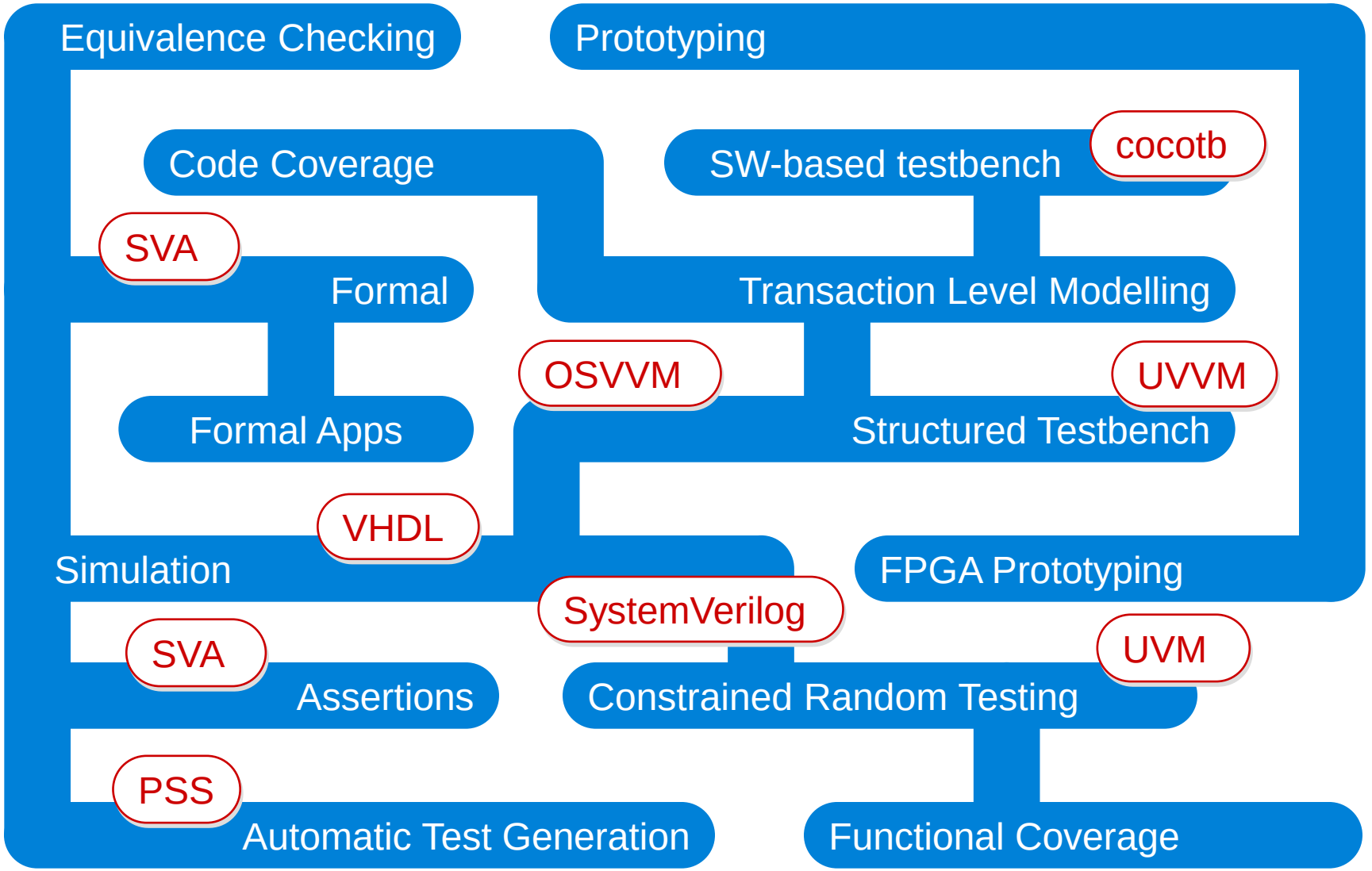
- In this course, we are concerned with Verification – how can we find design errors?
- This may be more difficult than doing the design itself



Widely accepted figure:
Verification accounts for ~70% of project effort

Copyright © 2015-2025 Doulos. All Rights Reserved

The Verification Maze



Copyright © 2015-2025 Doulos. All Rights Reserved

DUT

```
module wb_ram #(
  parameter de
  ...
) (
  input wb_clk
  input wb_rst
  input [31:0]
  input [31:0]
  input [3:0]
  input wb_we_
  input wb_cyc
  input wb_stb
  output reg w
  output reg [
  ...
);
```

```
entity wb_ram is
  generic (    depth : integer := 256  );
  port (
    wb_clk_i  : in std_logic;
    wb_rst_i  : in std_logic;
    wb_adr_i  : in std_logic_vector(31 downto 0);
    wb_dat_i  : in std_logic_vector(31 downto 0);
    wb_sel_i  : in std_logic_vector(3 downto 0);
    wb_we_i   : in std_logic;
    wb_cyc_i  : in std_logic;
    wb_stb_i  : in std_logic;
    wb_ack_o  : out std_logic;
    wb_dat_o  : out std_logic_vector(31 downto 0)
  );
end wb_ram;
```

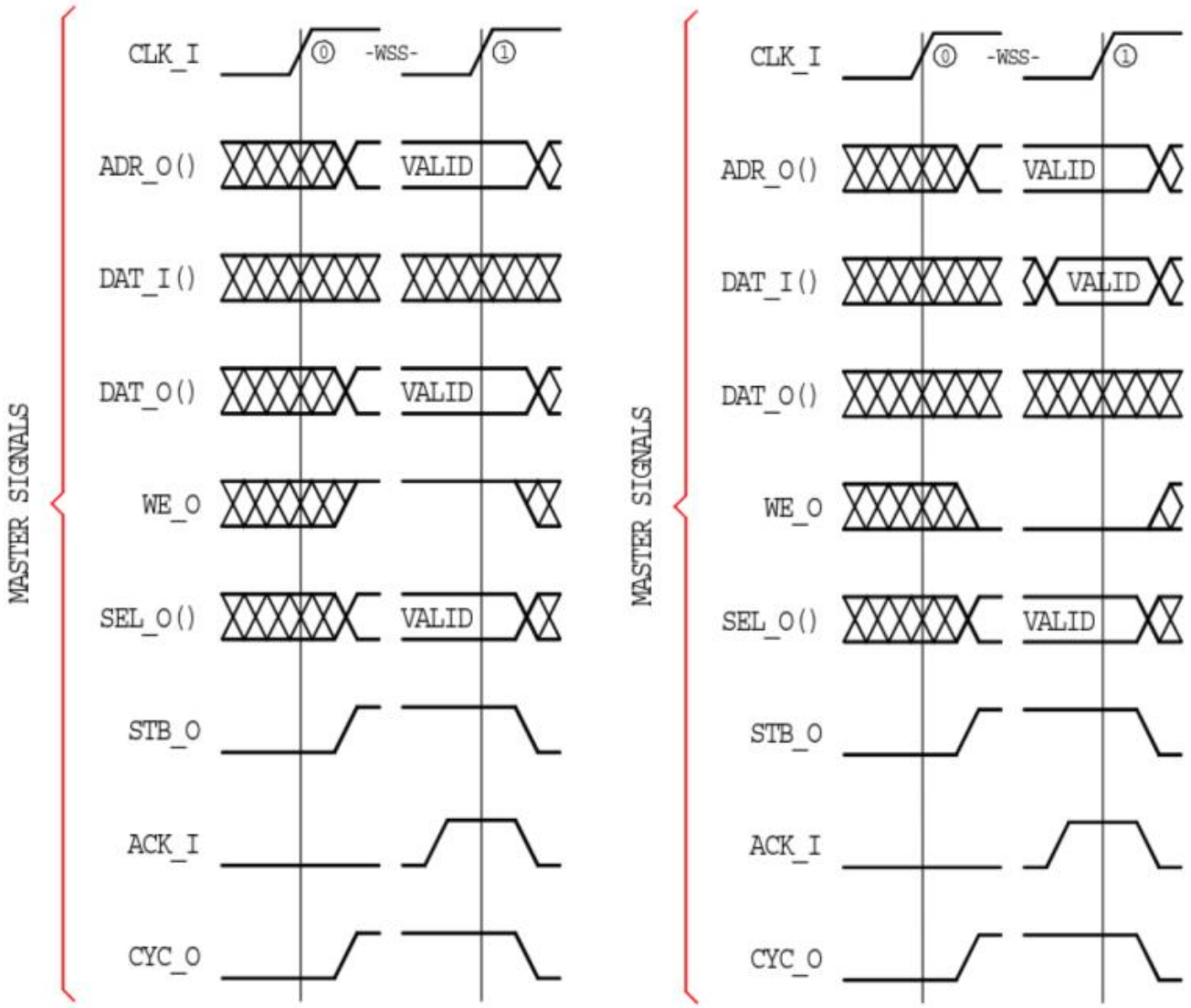
Copyright © 2015-2025 Doulos. All Rights Reserved

Licence


```
/* ISC License
 *
 * Verilog on-chip RAM with Wishbone interface
 *
 * Copyright (C) 2014, 2016 Olof Kindgren <olof.kindgren@gmail.com>
 *
 * Permission to use, copy, modify, and/or distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */
```

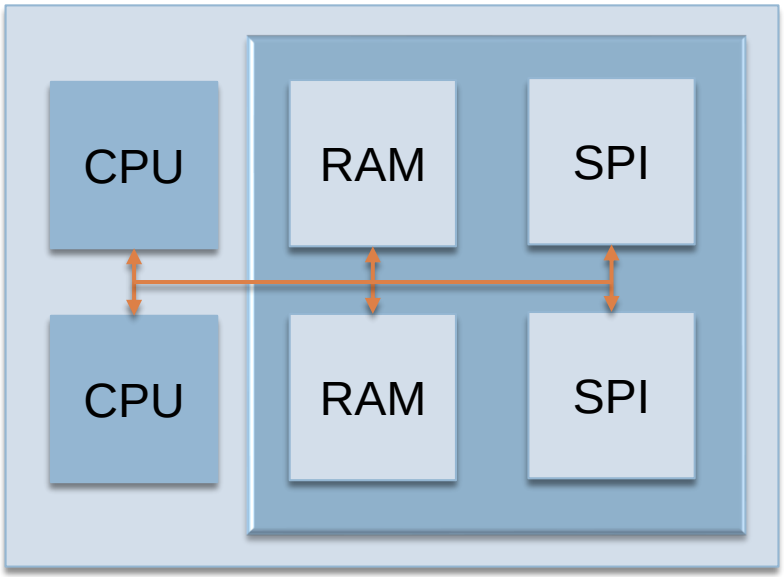
Copyright © 2015-2025 Doulos. All Rights Reserved

Wishbone Bus



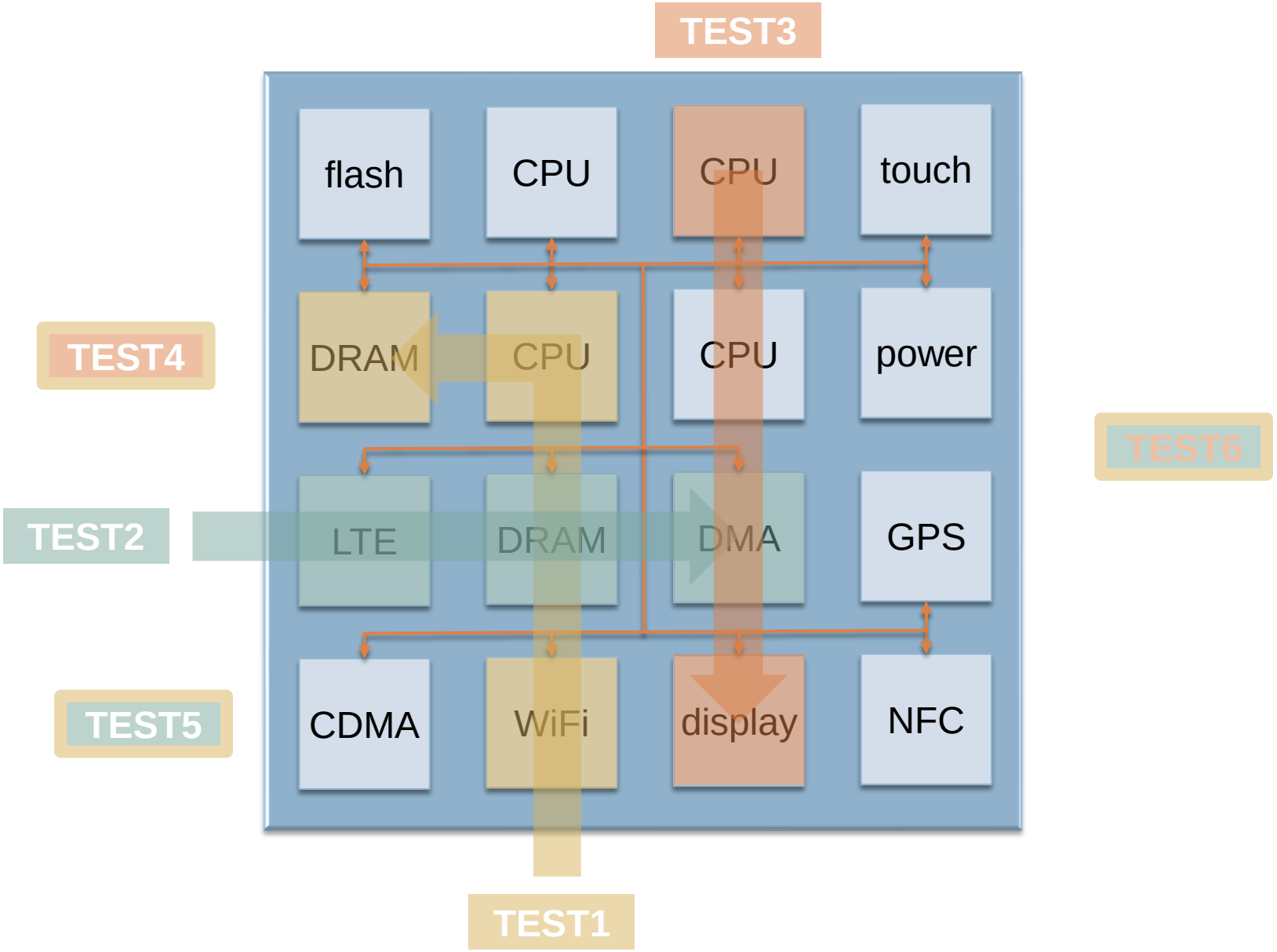
Copyright © 2015-2025 Doulos. All Rights Reserved

DUT



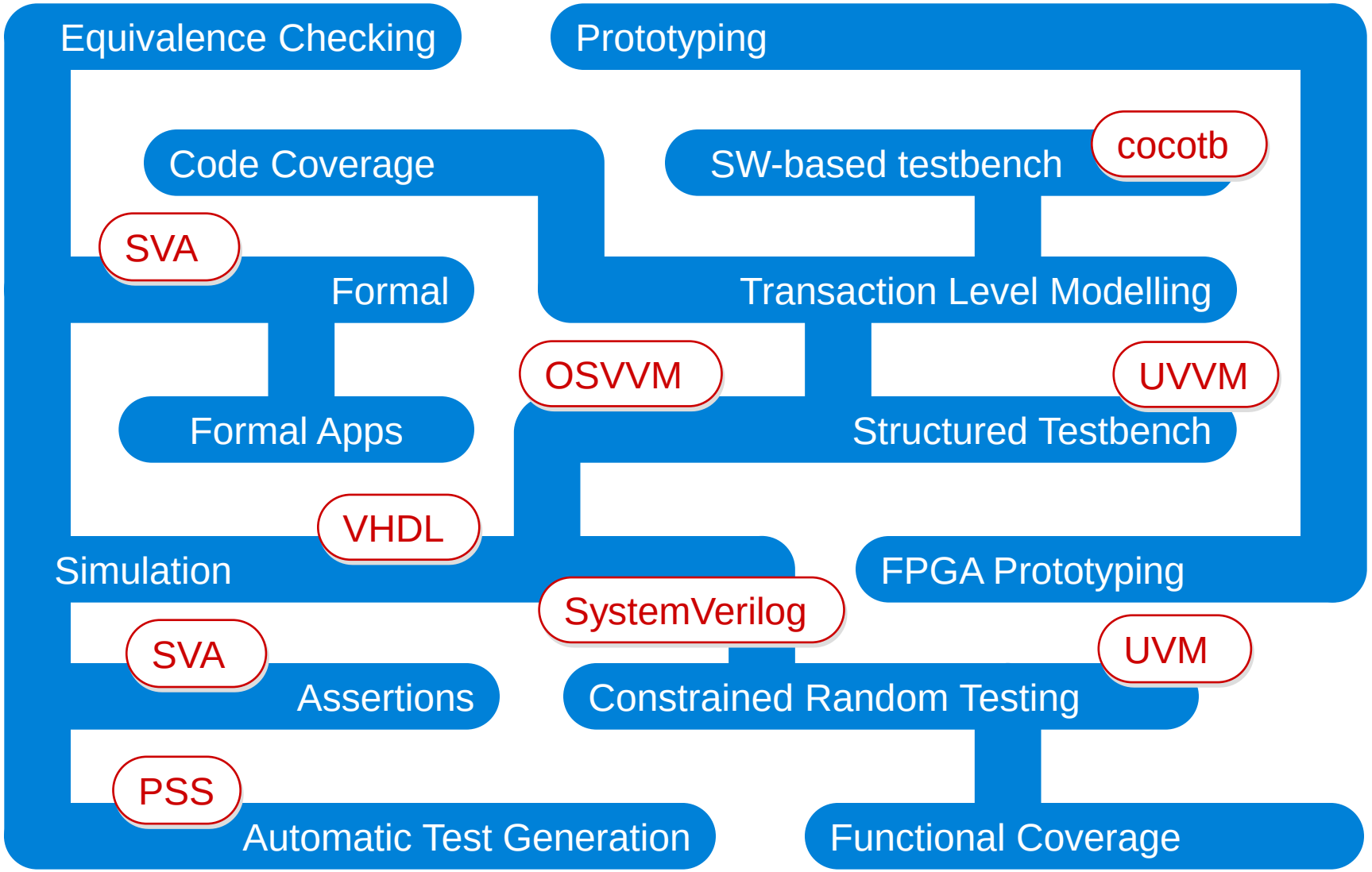
Copyright © 2015-2025 Doulos. All Rights Reserved

DUT



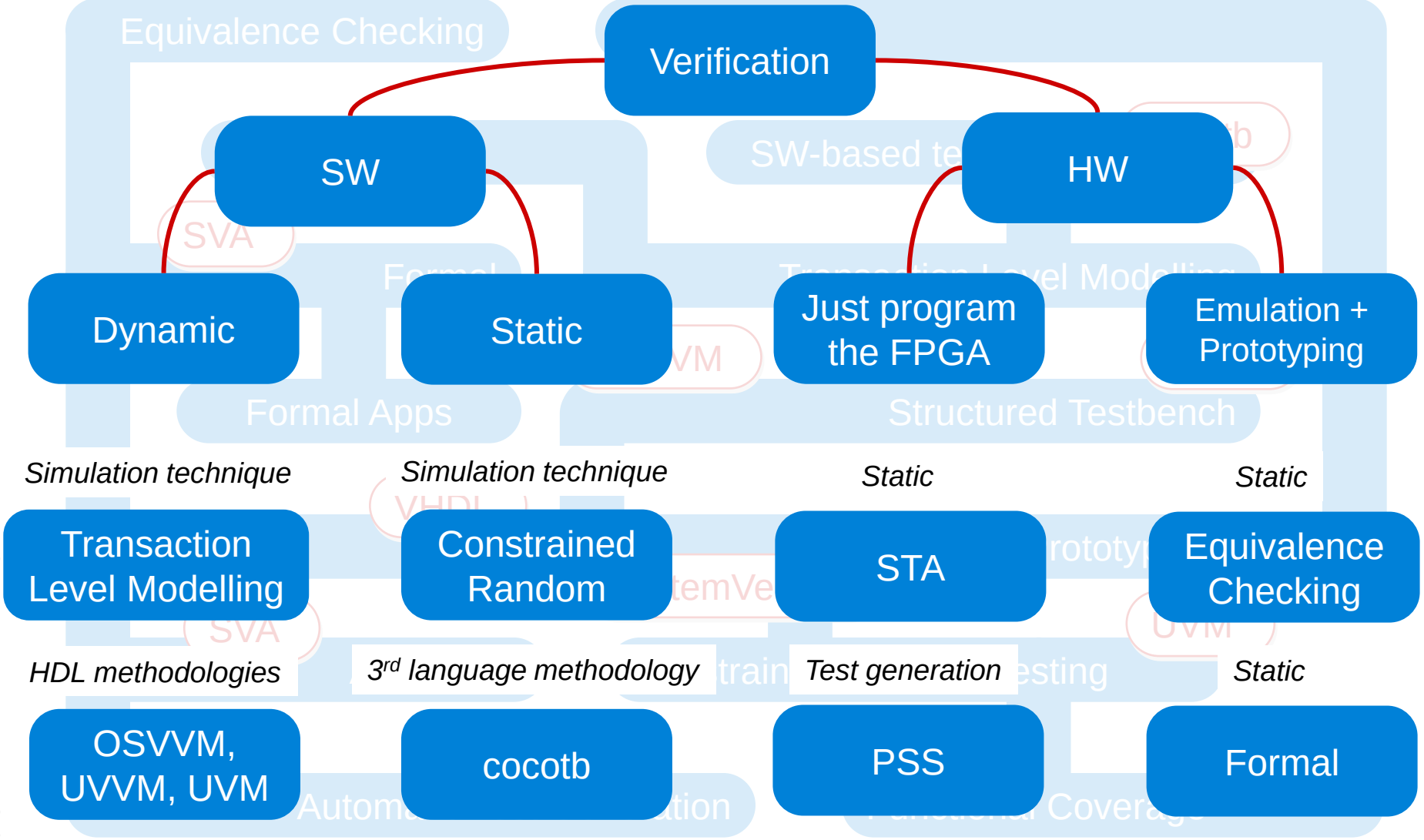
Copyright © 2015-2025 Doulos. All Rights Reserved

The Verification Maze



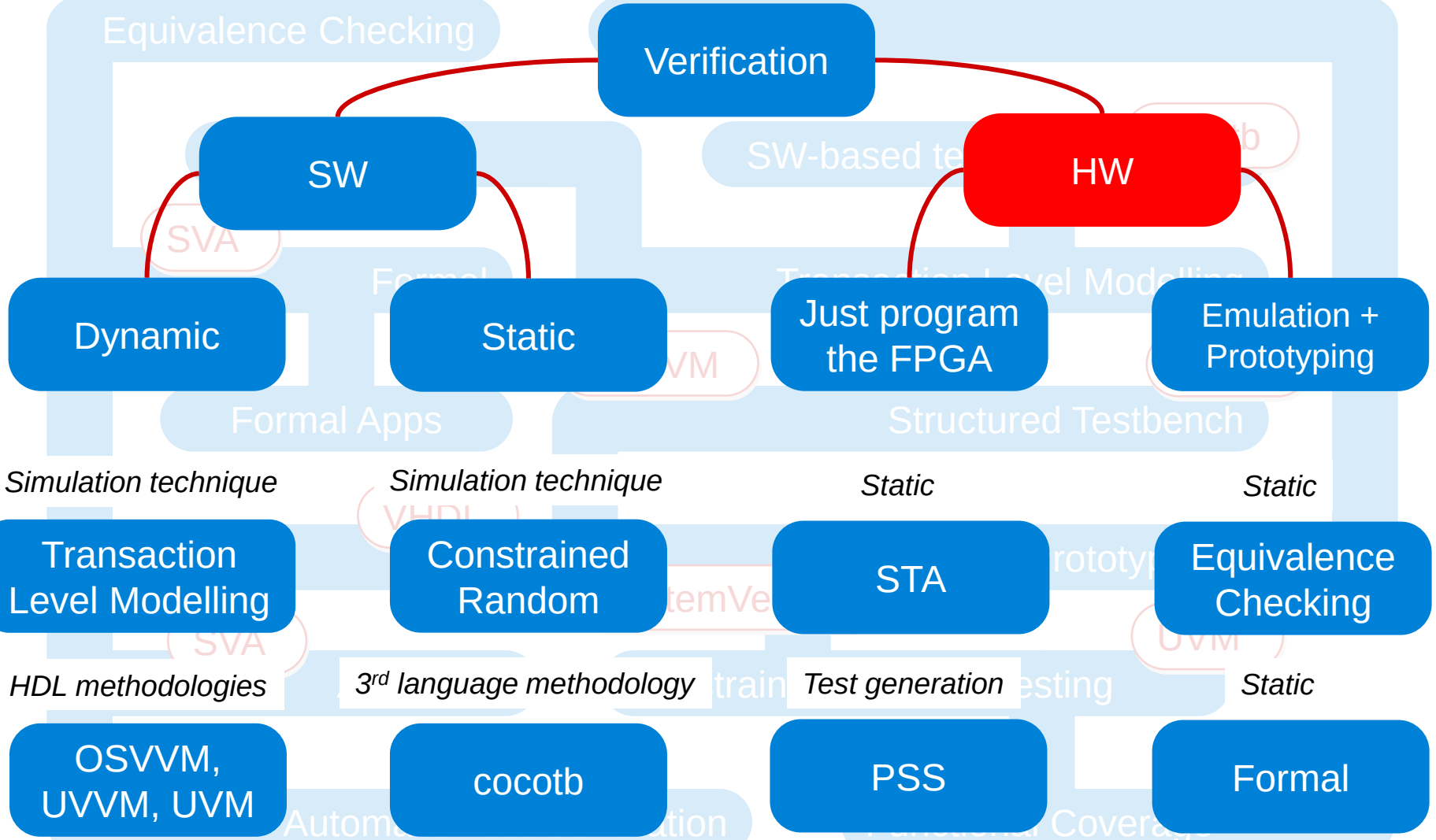
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



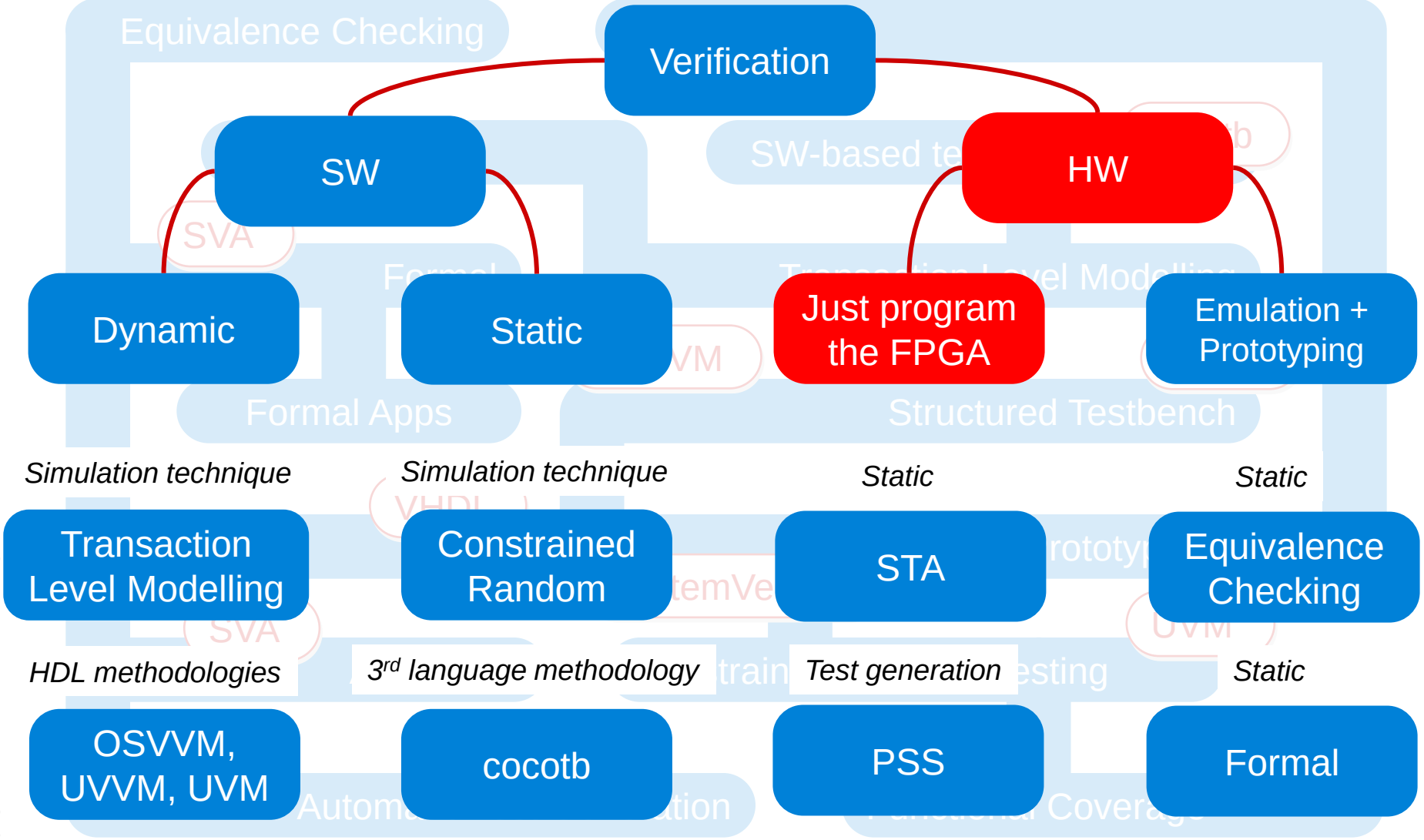
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



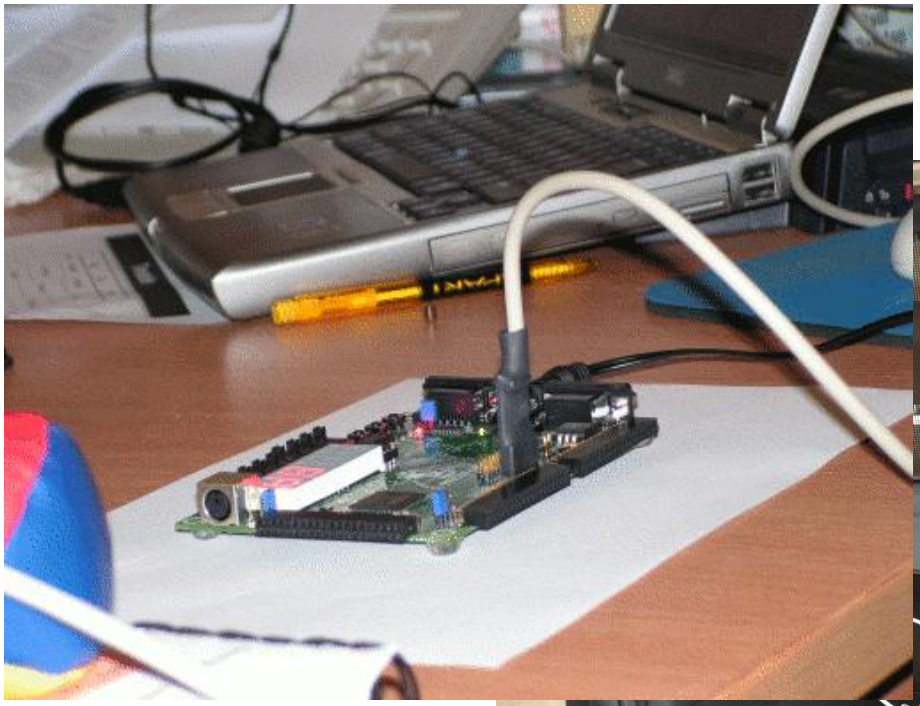
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



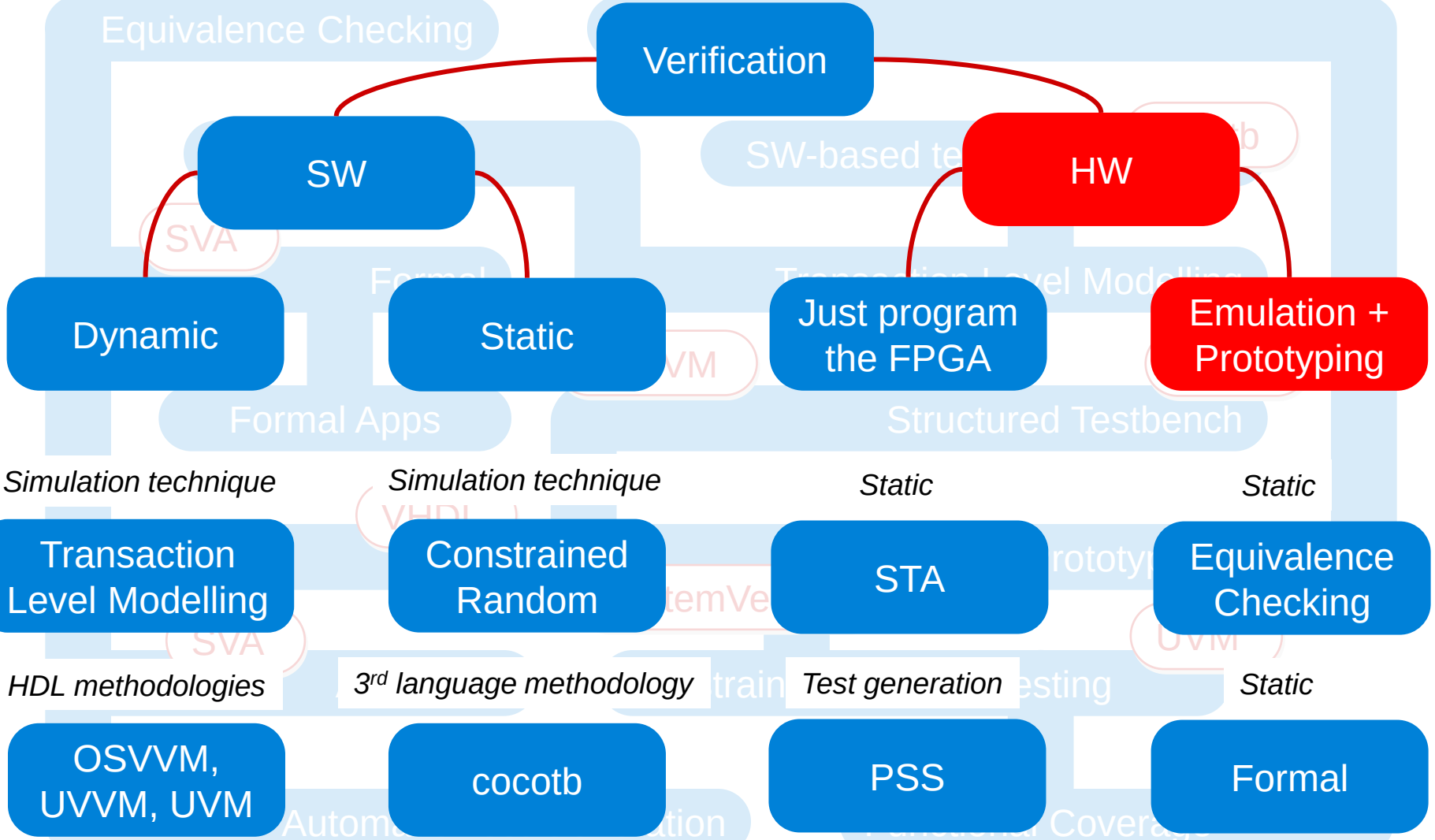
Copyright © 2015-2025 Doulos. All Rights Reserved

Just Program the FPGA



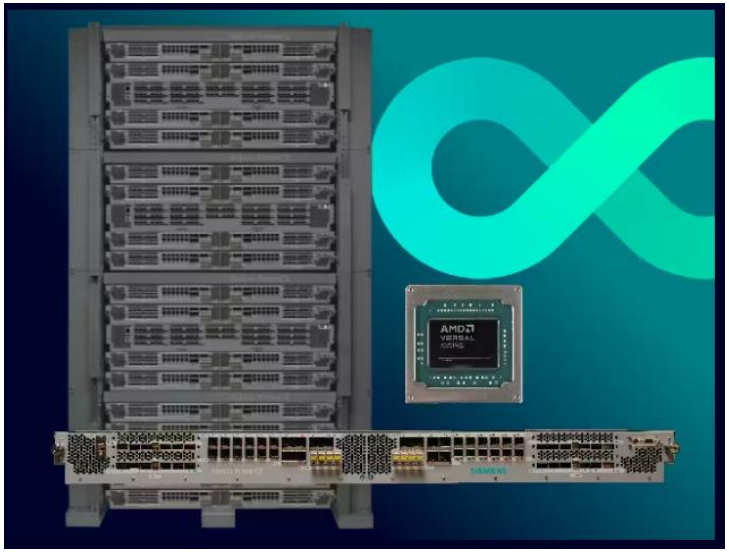
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



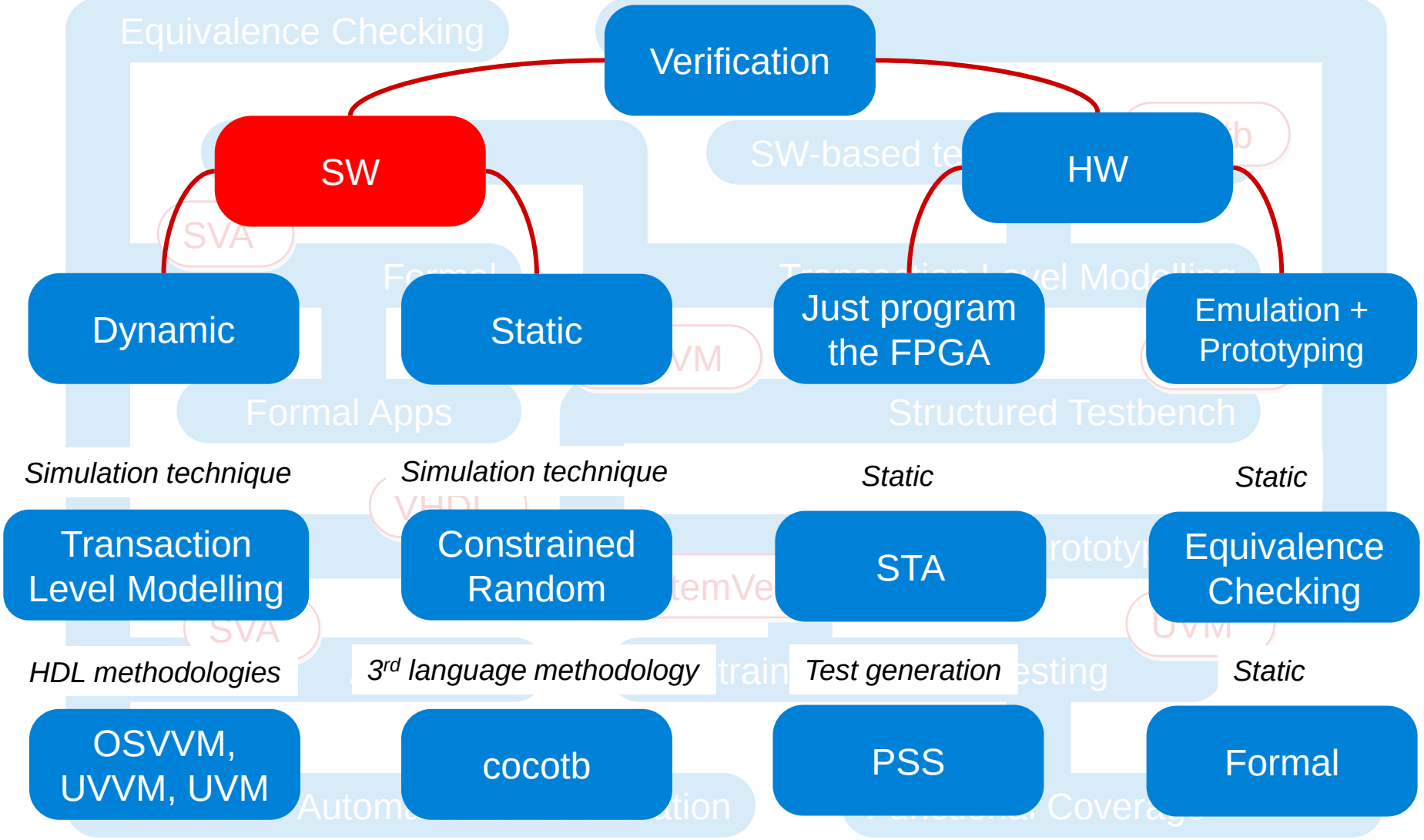
Copyright © 2015-2025 Doulos. All Rights Reserved

Emulation / Prototyping



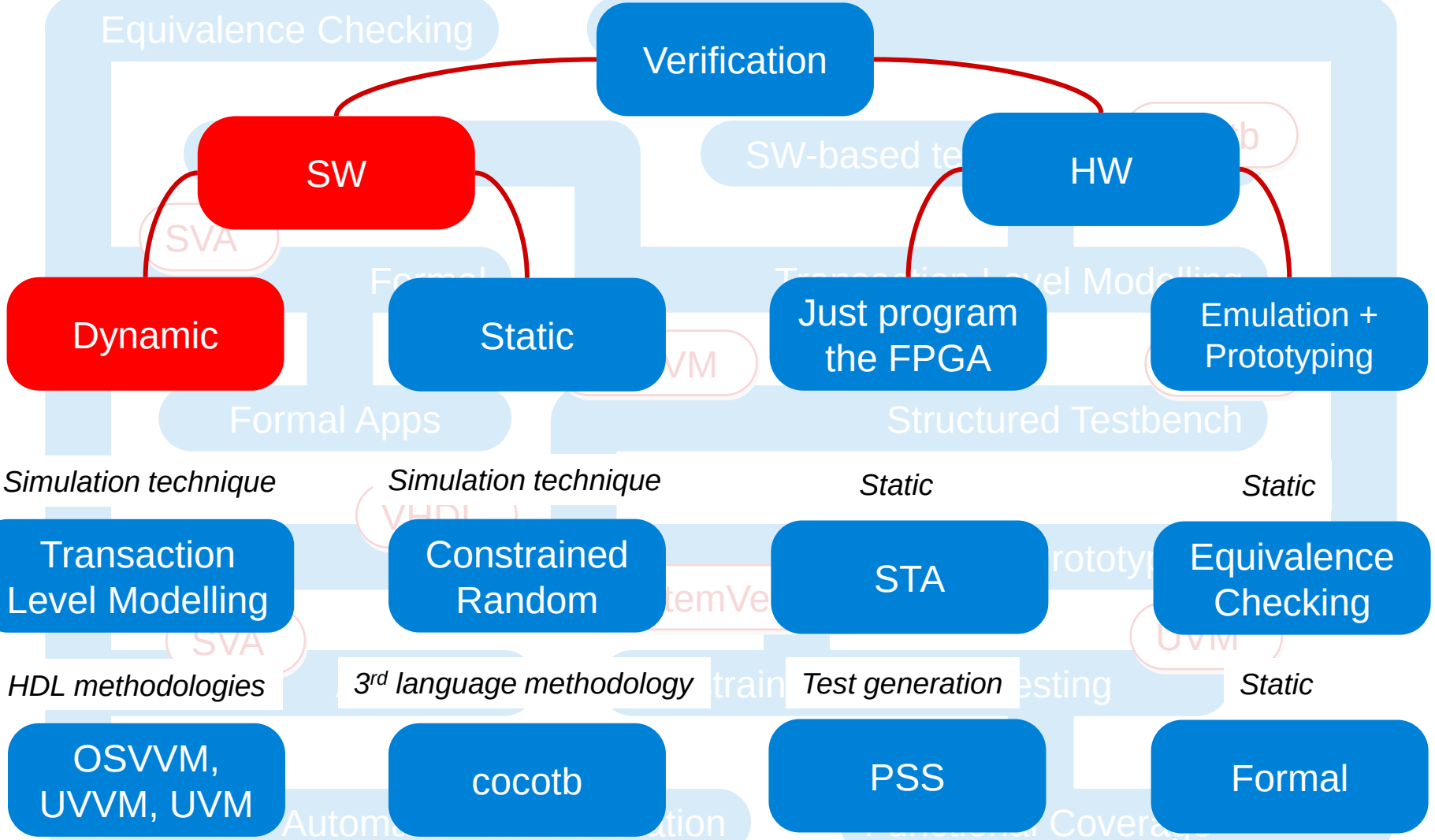
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

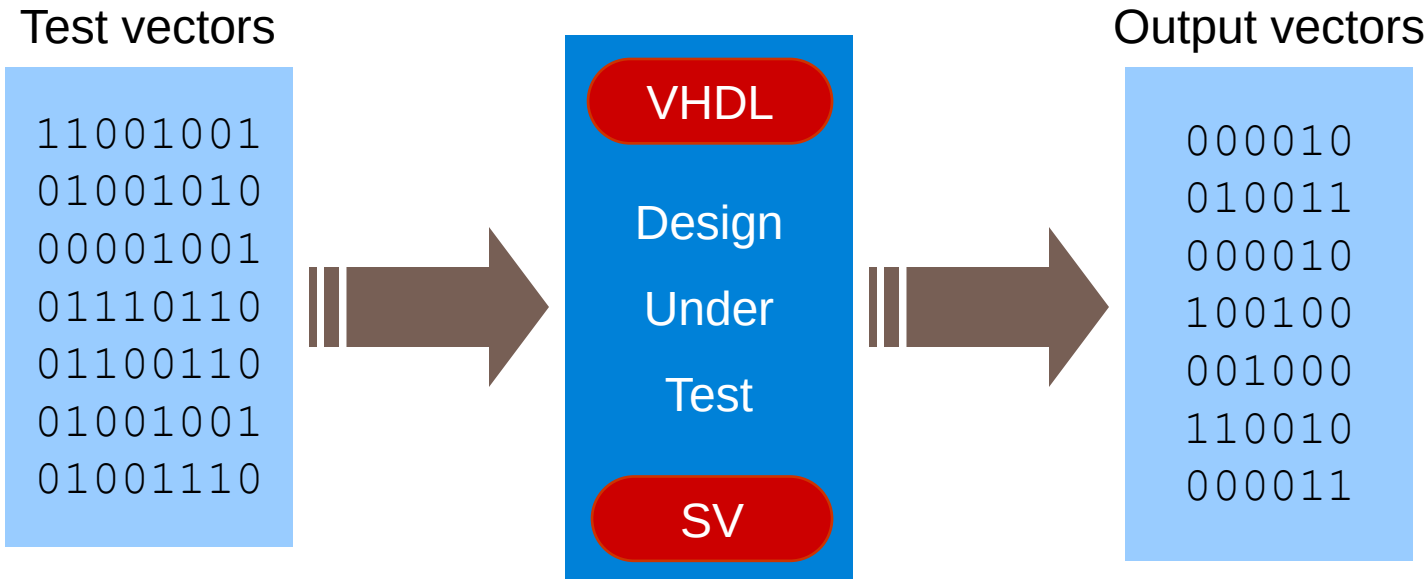
Simulation

- Execute a model of the system
- Most simulation is done at RTL (Register Transfer Level)
  - Simulate the same code which will be synthesized
- Cannot simulate everything – not enough time

- Widely used:
  - simulation is intuitively attractive, looks like the real thing
  - mature, familiar tools
  - excellent debugging

Copyright © 2015-2025 Doulos. All Rights Reserved

Simulating a Digital System



- Number one tool for functional verification
- Vectors must be created:
  - write a program to do it (“directed”)

  OR

  - compute them using constrained-random techniques (requires automated checking of output)

Copyright © 2015-2025 Doulos. All Rights Reserved

Basic Test Bench

```
initial begin
  wb_cyc = 1'b0; wb_stb = 1'b0;
  repeat (3)
  @(posedge wb_clk);

  // do a write
  @(posedge wb_clk); wb_cyc = 1'b0; wb_stb = 1'b0;
                     wb_adr = 32'h0; wb_dat = 32'h12345678; wb_sel = 4'b1111;
                     wb_we  = 1'b1;
  @(posedge wb_clk); wb_cyc = 1'b1; wb_stb = 1'b1;
  @(posedge wb_clk);

  // do a read
  @(posedge wb_clk); wb_cyc = 1'b0; wb_stb = 1'b0;
                     wb_adr = 32'h0; wb_dat = 32'hxxxxxxxx; wb_sel = 4'b1111;
                     wb_we  = 1'b0;
  @(posedge wb_clk); wb_cyc = 1'b1; wb_stb = 1'b1;
  ...
end

wb_ram #(.depth (MEMORY_SIZE)) dut ( ...
```

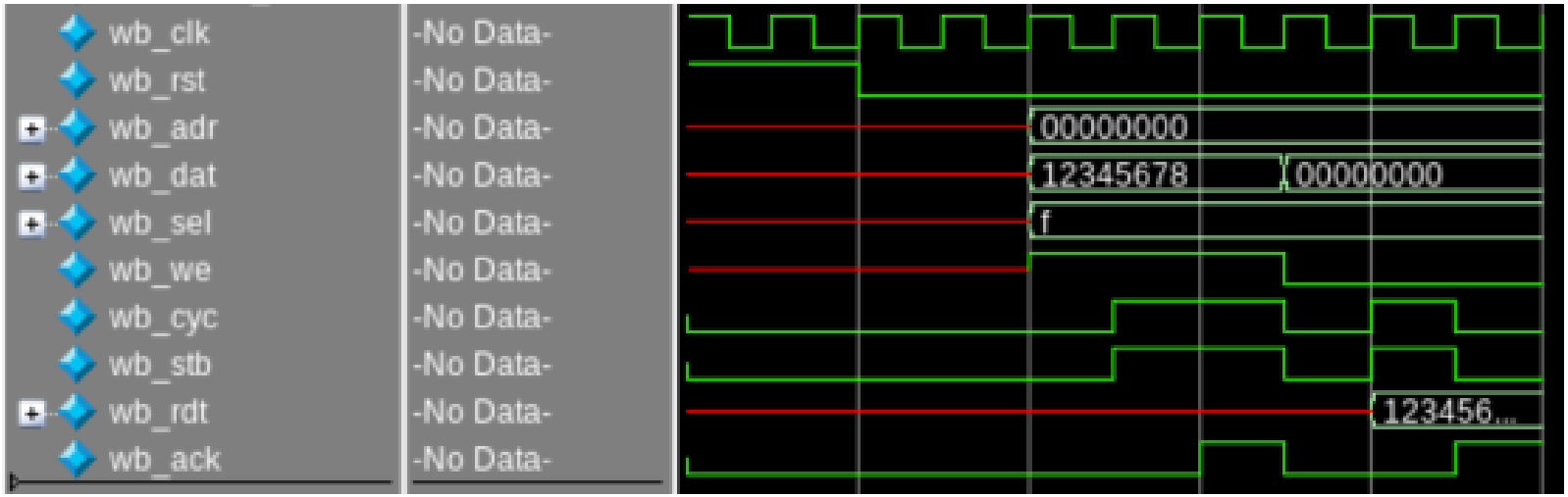
Copyright © 2015-2025 Doulos. All Rights Reserved

Basic Test Bench

```
process begin
  wb_cyc <= '0'; wb_stb <= '0';
  for I in 1 to 3 loop wait for rising_edge(wb_clk); end loop;

  -- do a write
  wait for rising_edge(wb_clk);
  wb_cyc <= '0'; wb_stb <= '0'; wb_we  = '1';
  wb_adr = (others => '0'); wb_dat <= X"12345678"; wb_sel <= "1111";
  wait for rising_edge(wb_clk);
  wb_cyc = '1'; wb_stb = '1';
  wait for rising_edge(wb_clk);

  -- do a read
  wait for rising_edge(wb_clk);
  wb_cyc <= '0'; wb_stb <= '0'; wb_we  = '0';
  wb_adr = (others => '0'); wb_dat <= (others => 'X')"; wb_sel <= "1111";
  wait for rising_edge(wb_clk);
  wb_cyc = '1'; wb_stb = '1';
  ...

dut: wb_ram generic map (depth => MEMORY_SIZE) port map ( ...
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Waveform Viewers



- Waveform viewers are a vital debugging tool
  - Supported by all major simulators
  - Display the value of design elements over a period of simulated time
  - Includes inputs, outputs and internal nodes
  - Can view scalar values and vectors

Copyright © 2015-2025 Doulos. All Rights Reserved

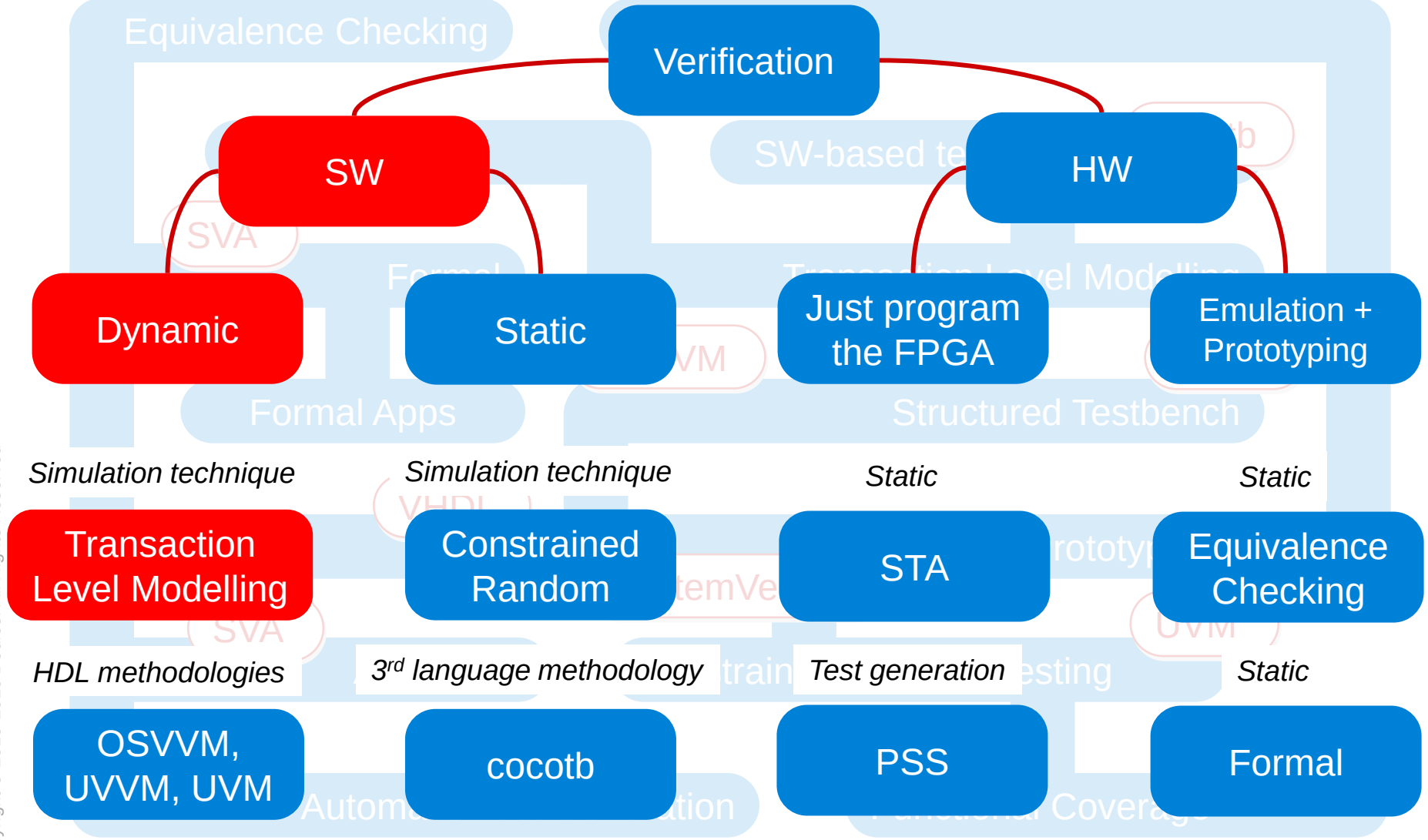
Self-Checking Test Benches

- Waveform-viewing is error-prone, time-consuming, not repeatable
- Self-checking is vital

```
...

// do a read
@(posedge wb_clk); wb_cyc = 1'b0; wb_stb = 1'b0;
                   wb_adr = 32'h0; wb_dat = 32'hxxxxxxxx;
                   wb_sel = 4'b1111;
                   wb_we  = 1'b0;
@(posedge wb_clk); wb_cyc = 1'b1; wb_stb = 1'b1;

// check the output data
@(posedge wb_clk); assert (wb_rdt === 32'h12345678);

...
```

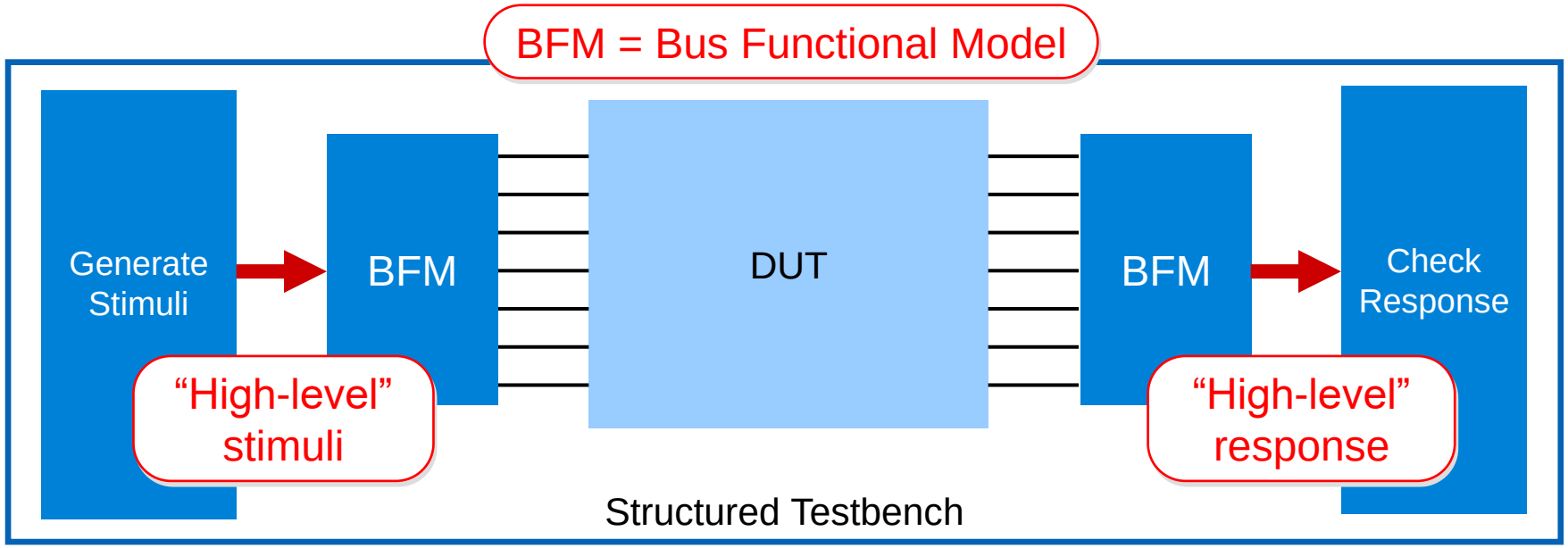
Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

Transaction Level Modelling



- **Transaction**: an atom of DUT behaviour
  - eg a read or write, a CPU instruction, a bus idle cycle

Copyright © 2015-2025 Doulos. All Rights Reserved

Transaction Level Modelling

```
task write (input [31:0] addr, data);
                    wb_cyc = 1'b0; wb_stb = 1'b0;
                    wb_adr = addr; wb_dat = data;
                    wb_sel = 4'b1111;
                    wb_we  = 1'b1;
  @(posedge wb_clk); wb_cyc = 1'b1; wb_stb = 1'b1;
  @(posedge wb_clk);
  @(posedge wb_clk);
endtask
```

write transaction

```
task read (input [31:0] addr, output [31:0] data);
                    wb_cyc = 1'b0; wb_stb = 1'b0;
                    wb_adr = addr; wb_dat = 32'hxxxxxxxx;
                    wb_sel = 4'b1111;
                    wb_we  = 1'b0;
  @(posedge wb_clk); wb_cyc = 1'b1; wb_stb = 1'b1;
  @(posedge wb_clk); data = wb_rdt;
endtask
```

read transaction

Copyright © 2015-2025 Doulos. All Rights Reserved

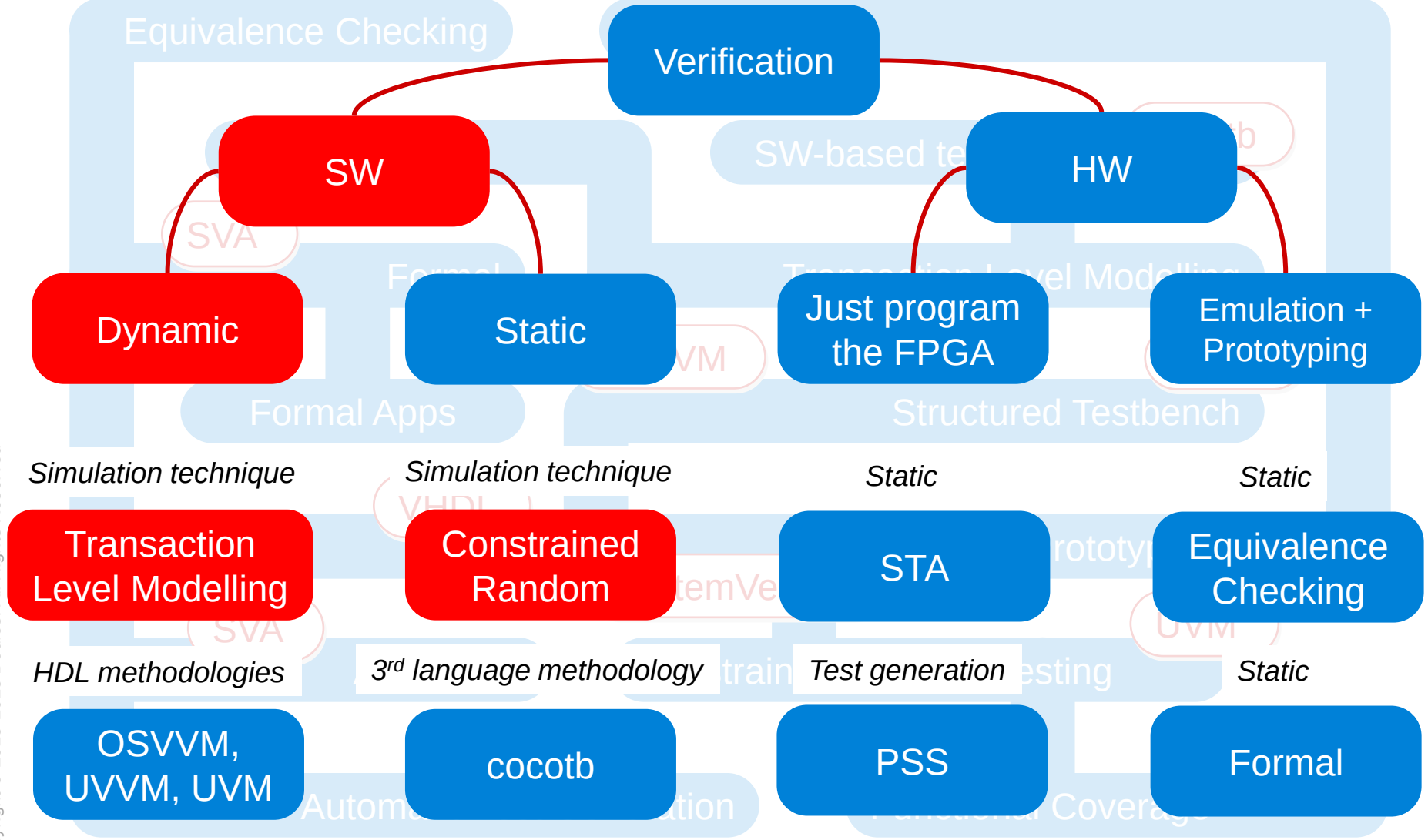
Transaction Level Modelling

```
logic [31:0] actual_data;

initial begin
  ...
  // do a write
  write(32'h0, 32'hdeadbeef );

  // do a read
  read(32'h0, actual_data);

  // check the output data
  assert (actual_data === 32'hdeadbeef);
  ...
  $stop;
end
```

the “test”

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



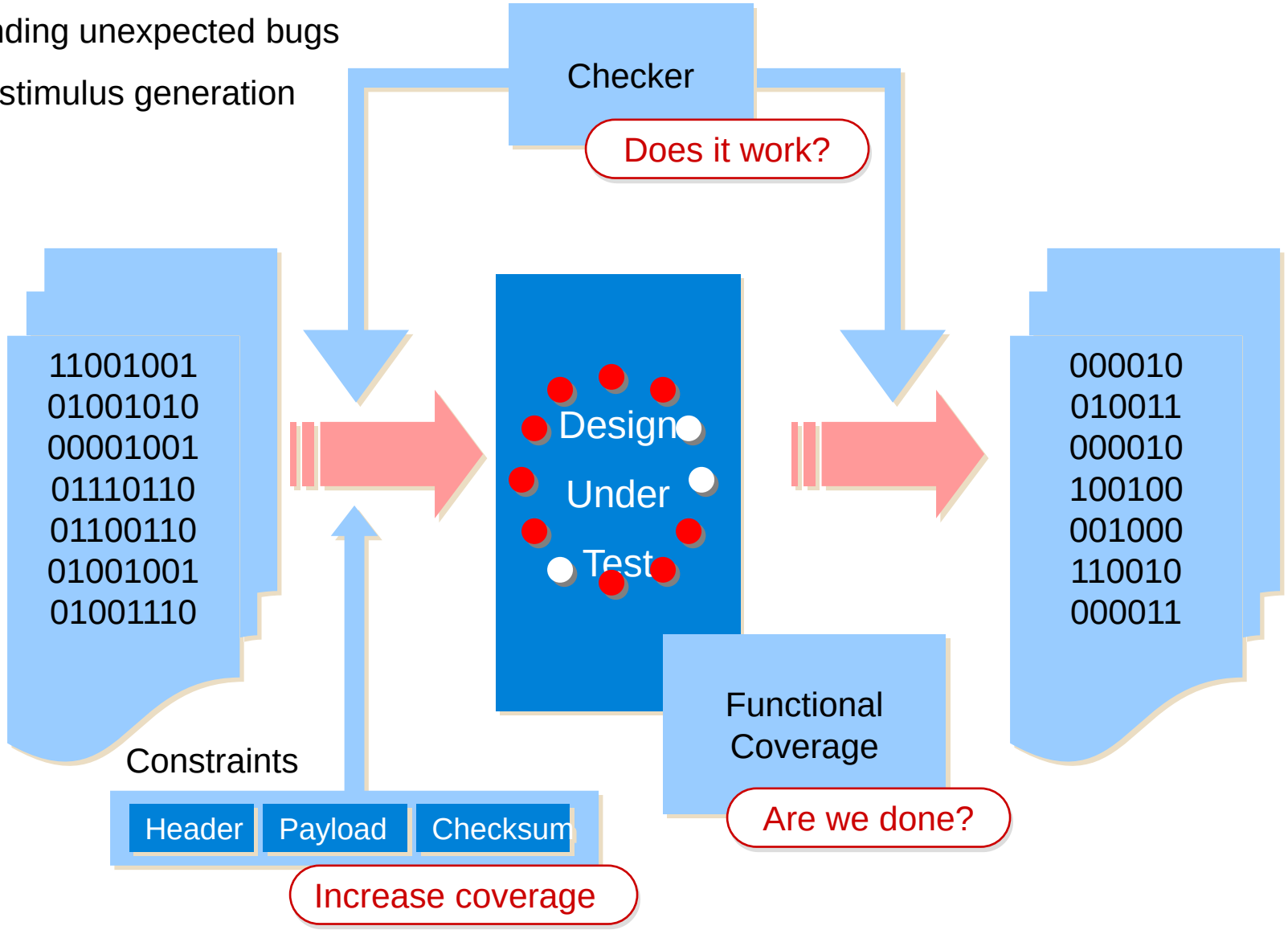
Copyright © 2015-2025 Doulos. All Rights Reserved

Constrained Random Verification

Constrained random stimulus

Good for finding unexpected bugs

Automates stimulus generation



Copyright © 2015-2025 Doulos. All Rights Reserved

Constrained Random

```
logic [31:0] addr;
logic [31:0] mem [0:MEMORY_SIZE-1];

for (int i = 32'b0; i < 10000; i = i + 4) begin

  addr = $urandom_range(0, (MEMORY_SIZE/4)-1) << 2;

  if ($urandom_range(0,1) == 1) begin
    mem[addr] = $urandom;
    write(addr, mem[addr]);
  end else begin
    read(addr, actual_data);
    assert ((actual_data==mem[addr]) || $isunknown(mem[addr]))
      else
        $display("ERROR ... );
  end

end
```

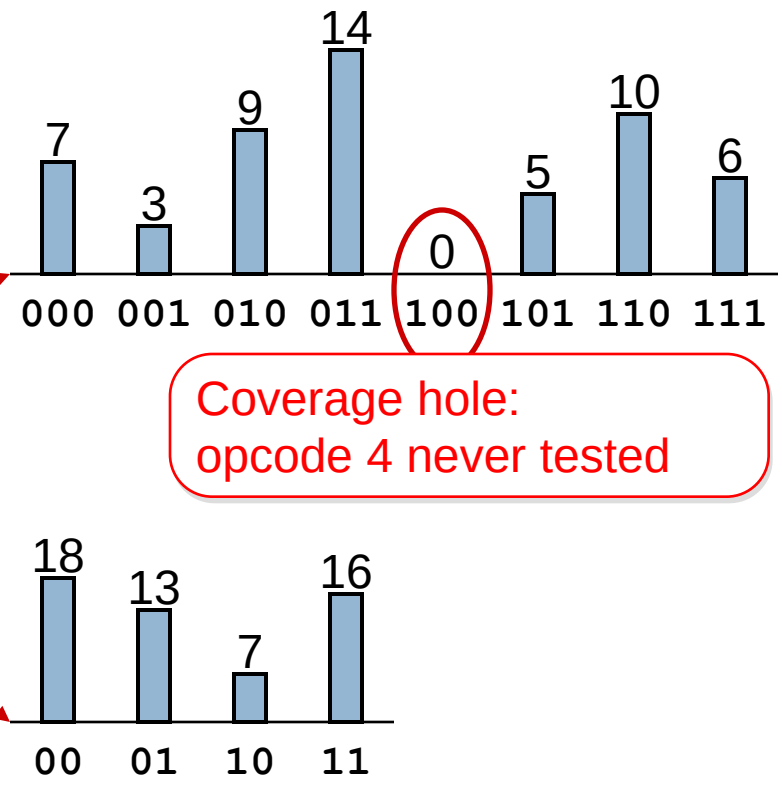
Copyright © 2015-2025 Doulos. All Rights Reserved

Coverage Data Collection

- Every coverage item has a set of *bins*
- One bin for each possible value of the item
- Each bin acts as a counter
  - Incremented on each sampling event when the item has the corresponding value

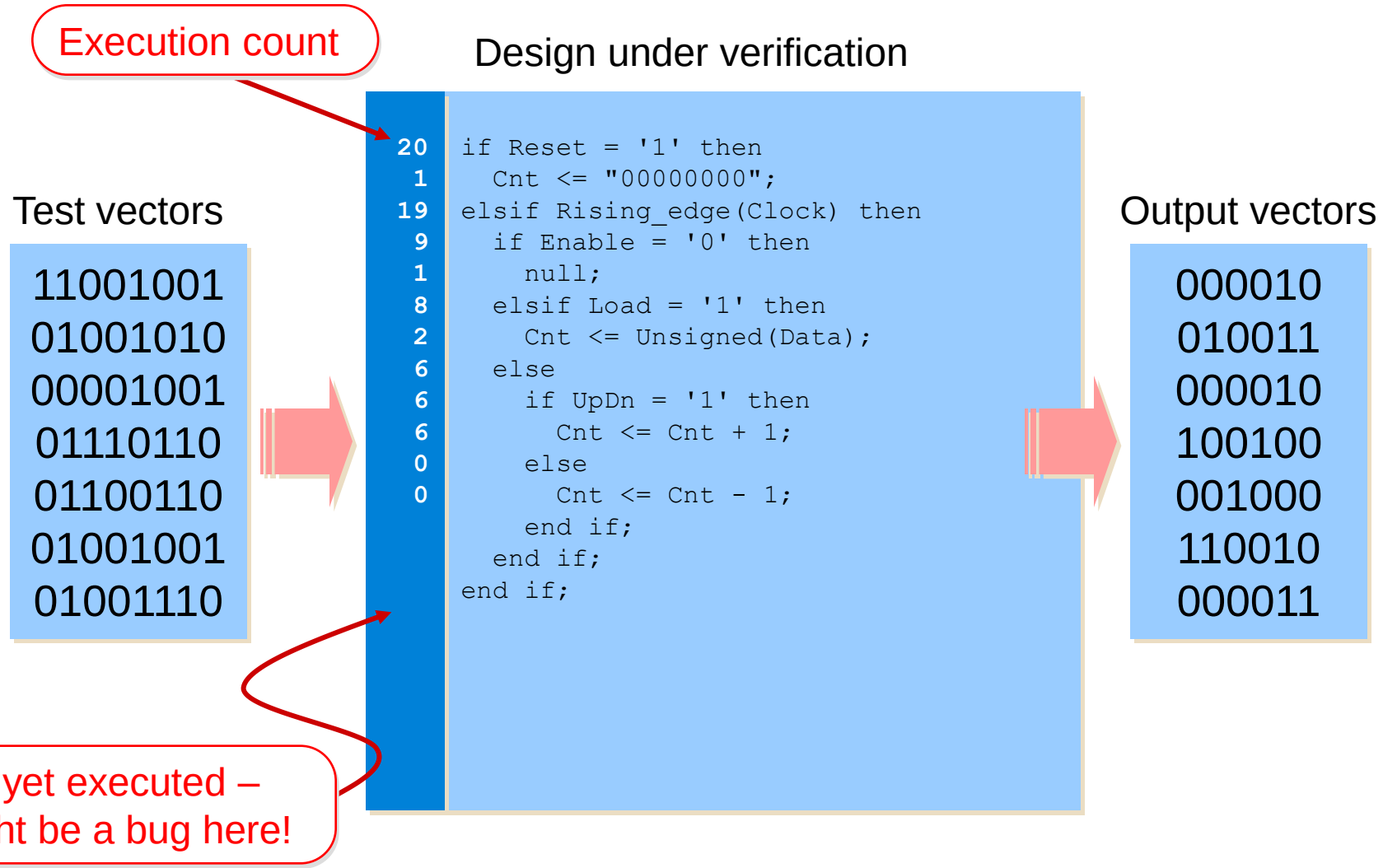
```
reg [2:0] Opcode;
reg [1:0] AdrsMode;
...
covergroup G @(posedge Clk);
  coverpoint Opcode;
  coverpoint AdrsMode;
endgroup
```

SystemVerilog



Copyright © 2015-2025 Doulos. All Rights Reserved

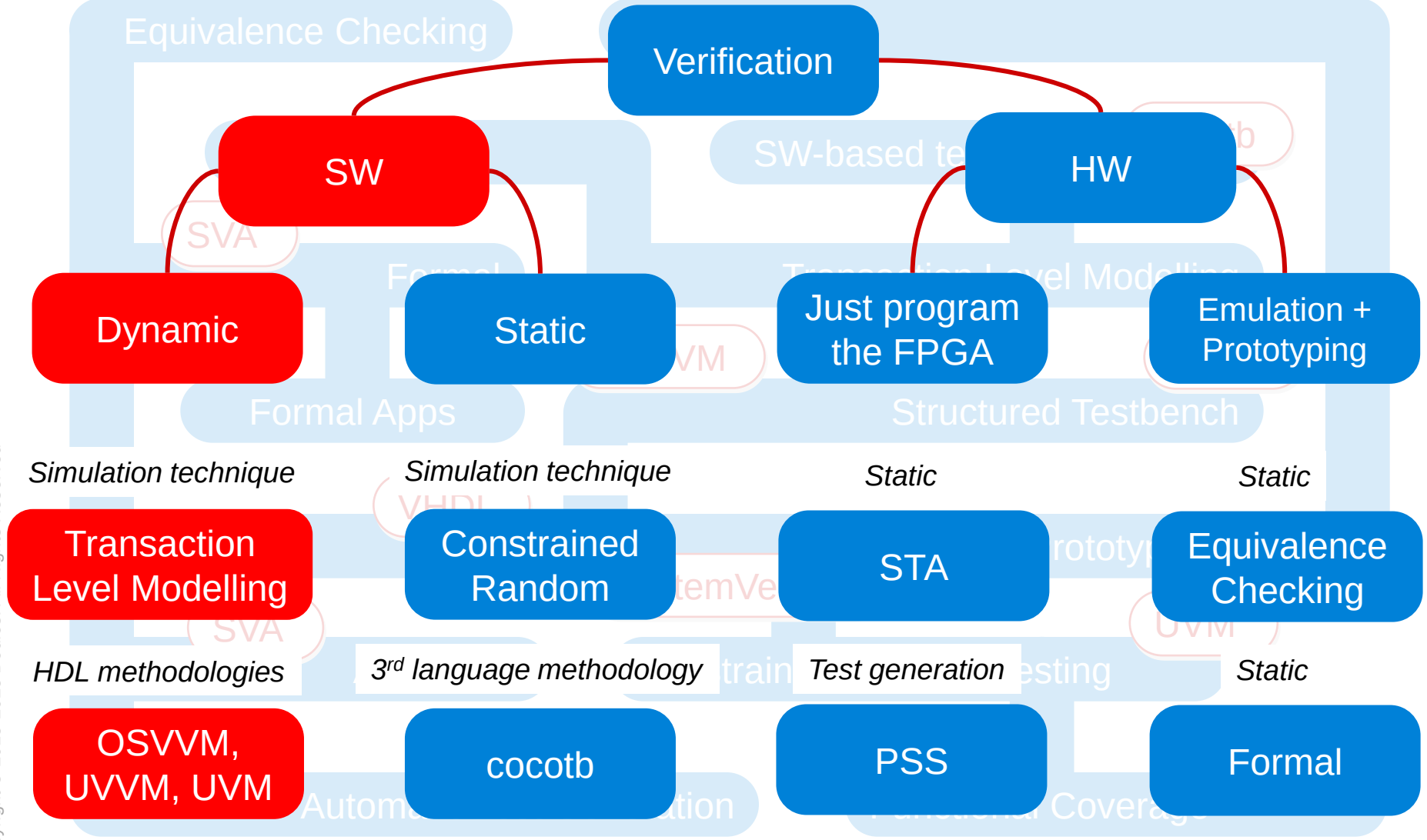
Code Coverage



- Doesn't prove correctness!

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



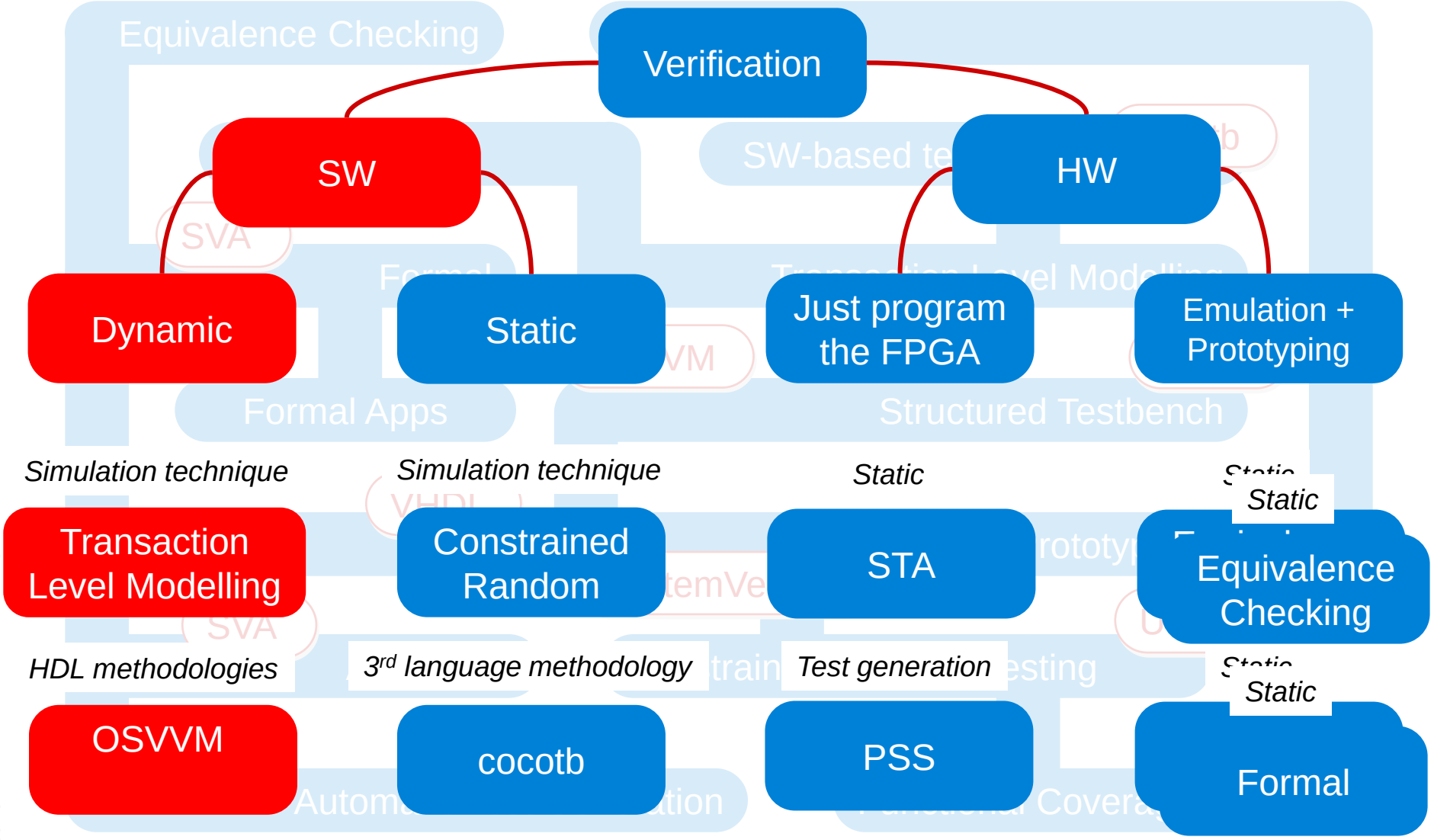
Copyright © 2015-2025 Doulos. All Rights Reserved

Methodology

```
...
vlog_tb_utils vlog_tb_utils0();
vlog_tap_generator #("wb_ram.tap", 1) vtg();
wb_bfm_transactor #(...) master (.wb_clk_i (wb_clk), ...

wb_ram #(...) dut (.wb_clk_i (wb_clk), .wb_rst_i (wb_rst), ...

integer TRANSACTIONS, SUBTRANSACTIONS, SEED;
initial begin
  if($value$plusargs("transactions=%d", TRANSACTIONS))
    master.set_transactions(TRANSACTIONS);
    ...
    master.display_settings;
    master.run;
    master.display_stats;
 end
...
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

What is OSVVM?

OSVVM, the Open Source VHDL Verification Methodology

- Developed by SynthWorks Design Inc and Aldec Inc
- Free and Open Source

- VHDL verification framework
- Verification utility library
  - Randomization, functional coverage, error logging etc
- Verification component library
  - AXI4, Ethernet, SPI and Wishbone etc
- Scripting API
- Co-simulation capability

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Utility Library

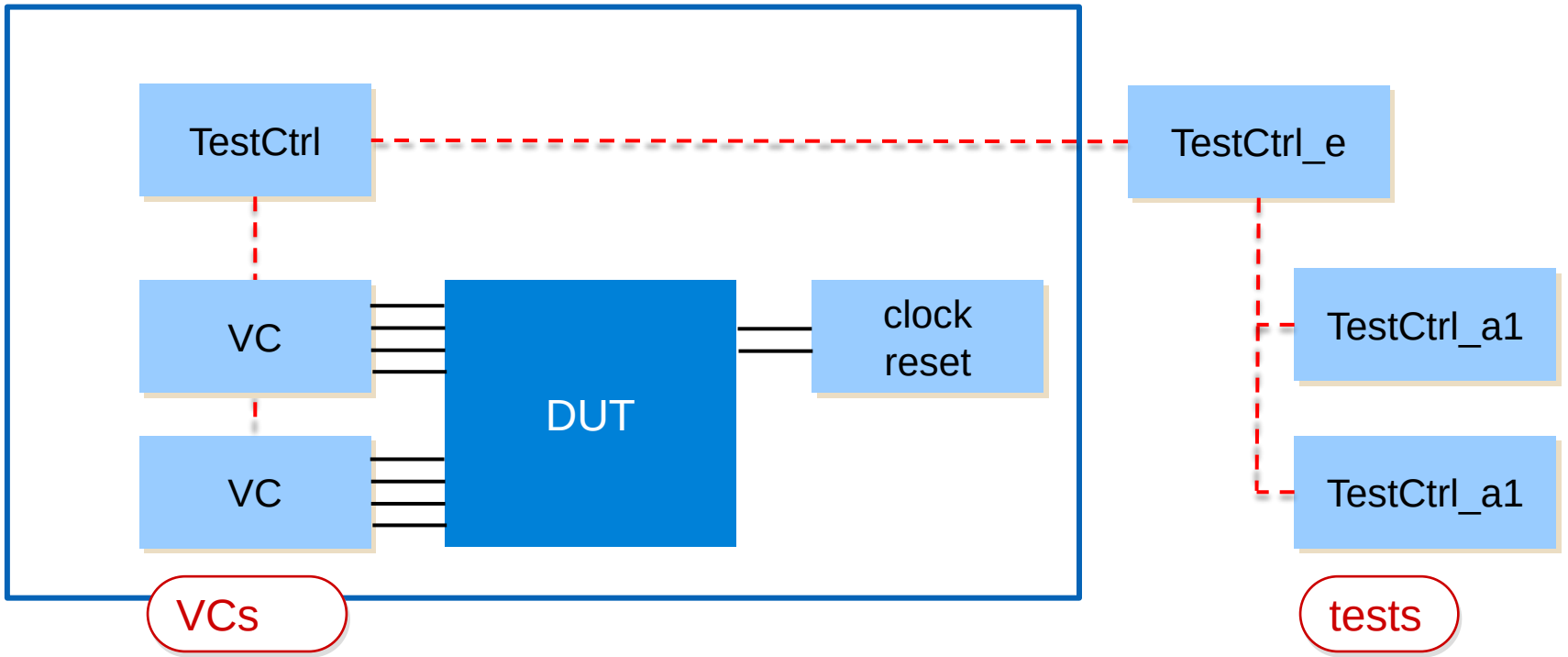
- Collection of useful testbench utility functions and procedures

Category	Example
Constrained Random testing	`RandomPkg`
Functional Coverage	`CoveragePkg`
Error logging and reporting	`AlertLogPkg`
Useful testbench utilities	`TbUtilPkg`
Clock and Reset control	`ClockResetPkg`
Common transcript file across a testbench	`TranscriptPkg`
A generic general-purpose scoreboard	`ScoreboardGenericPkg`
API/data structure for Memory models	`MemoryPkg`
Resolved types used by OSVVM	`ResolutionPkg`
Basic string handling and reading utilities	`TextUtilPkg`

Copyright © 2015-2025 Doulos. All Rights Reserved

OSVVM Structured Testbench Framew

- Implements a structured testbench
- VC = Verification Component



Copyright © 2015-2025 Doulos. All Rights Reserved

OSVVM – Transactions

- Two standard transaction record types

```
type StreamRecType is record
  Rdy              : RdyType ;
  Ack              : AckType ;
  Operation        : StreamOperationType ;
  DataToModel      : std_logic_vector_max_c ;
  ParamToModel     : std_logic_vector_max_c ;
  DataFromModel    : st
  ParamFromModel   : st
  ...
  end record StreamRec
```

```
type AddressBusRecType is record
  Rdy              : bit_max;
  Ack              : bit_max;
  Operation        : AddressBusOperationType ;
  Address          : std_logic_vector_max_c;
  AddrWidth        : integer_max;
  DataToModel      : std_logic_vector_max_c;
  DataFromModel    : std_logic_vector_max_c;
  ...
end record AddressBusRecType;
```

Copyright © 2015-2025 Doulos. All Rights Reserved

OSVVM Test Harness

```
package Manager is new OSVVM_WISHBONE.WishboneGenericSignalsPkg
  generic map (
    ADDR_WIDTH   => 32,
    DATA_WIDTH   => 32
  ) ;
...
DUT_1 : wb_ram generic map ( ... ) port map (
  wb_clk_i    => Clk, wb_rst_i    => not nReset,
  wb_adr_i    => Manager.WishboneBus.Adr,
  ...
  wb_dat_o    => Manager.WishboneBus.iDat, ... );

CreateClock ( Clk => Clk, Period => Tperiod_Clk );
CreateReset ( Reset => nReset, ResetActive => '0', Clk => Clk, ...

Manager_1 : WishboneManager port map (
  Clk          => Clk,
  nReset       => nReset,
  WishboneBus  => Manager.WishboneBus,
  TransRec     => Manager.TransRec
);
...
```

Wishbone signals

DUT

VC

```
TestCtrl_1 : TestCtrl port map (
    nReset       => nReset,
    ManagerRec   => Manager.TransRec
);
```

Test Sequencer

Copyright © 2015-2025 Doulos. All Rights Reserved

OSVVM Test Sequencer – Test

```
architecture BasicReadWrite of TestCtrl is
begin

  ControlProc : process
  begin
    SetLogEnable(PASSED, TRUE);
    SetLogEnable(INFO, TRUE);
    wait for 0 ns;  wait for 0 ns;
    TranscriptOpen;
    SetTranscriptMirror(TRUE);

    wait until nReset = '1';
    WaitForBarrier(OsvvmTestDone, 35 ms);

    TranscriptClose;

    EndOfTestReports(TimeOut => now >= 35 ms);
    std.env.stop;
    wait;
  end process ControlProc;
```

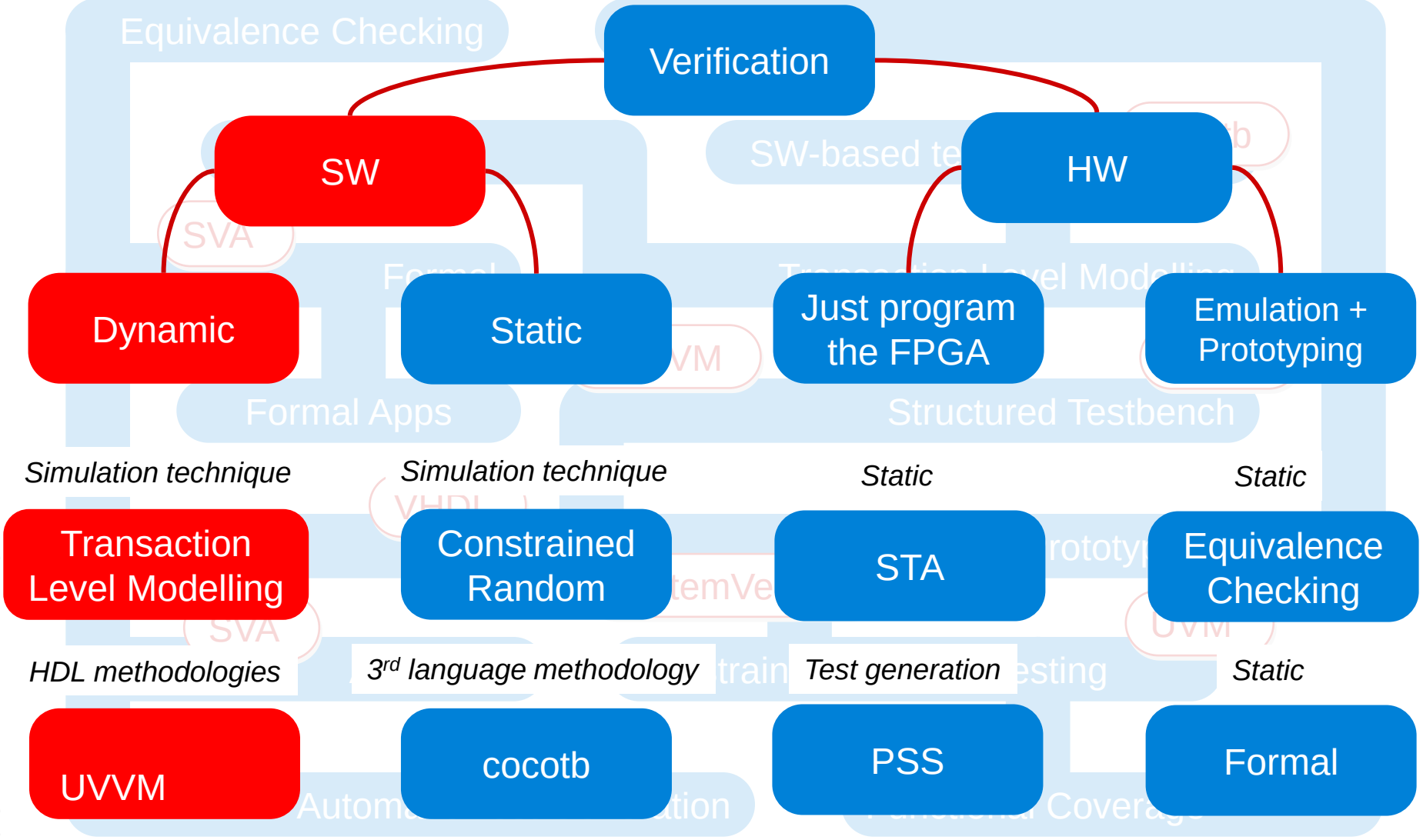
Copyright © 2015-2025 Doulos. All Rights Reserved

OSVVM Test Sequencer – Test


```
  ManagerProc : process
    ...
  begin
    wait until nReset = '1';
    WaitForClock(ManagerRec, 2);
    ID := NewID("Tb");

    log(ID, "Write, Read, ReadCheck words, 4 Bytes");
    Addr := ADDR_ZERO;
    Write(ManagerRec,   Addr, X"5555_5555" );
    Read(ManagerRec,    Addr, Data);
    AlertIfNotEqual(ID, Data, X"5555_5555", "Read Data: ");

    ...

    WaitForClock(ManagerRec, 2);
    WaitForBarrier(OsvvmTestDone);
    wait;
  end process ManagerProc;

end BasicReadWrite;
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

What is UVVM?

UVVM, the Universal VHDL Verification Methodology

- Developed by Emlogic
  - Released under MIT License
- Two parts:
  - VVC Framework
  - Utility Library

Copyright © 2015-2025 Doulos. All Rights Reserved

Utility Library

- Collection of useful testbench utility functions and procedures

Category	Example
Logging and Verbosity Control	`disable_log_msg(msg_id, …)`
Alert Handling	`set_alert_file_name(…)`
Checks and Awaits	`await_change(…)`
String Handling	`to_string(…)`
Randomization	`randomize(…)`
Signal Generators	`gen_pulse(…)`
Synchronization	`await_barrier(…)`
BFM Common Package	`wait_num_rising_edge(…)`

```
library uvvm_util;
  context uvvm_util.uvvm_util_context;
```

Copyright © 2015-2025 Doulos. All Rights Reserved

UVVM Test Harness

```
...
i_ti_uvvm_engine : entity uvvm_vvc_framework.ti_uvvm_engine;

DUT_1 : wb_ram generic map ( ... ) port map (
  wb_clk_i   => clk, wb_rst_i   => arst,
  wb_adr_i   => wishbone_vvc_master_if.adr_o,
  wb_dat_i   => wishbone_vvc_master_if.dat_o,
  ...
  wb_dat_o   => wishbone_vvc_master_if.dat_i,, ... );

i1_wishbone_vvc : entity bitvis_vip_wishbone.wishbone_vvc
  generic map ( ... ) port map (
    clk                    => clk,
    wishbone_vvc_master_if => wishbone_vvc_master_if );

p_arst : arst <= '1', '0' after 5 * C_CLK_PERIOD;

i_clock_generator_vvc : entity bitvis_vip_clock_generator.clock_generator_vvc
  generic map( ... ) port map(
    clk => clk );
...
```

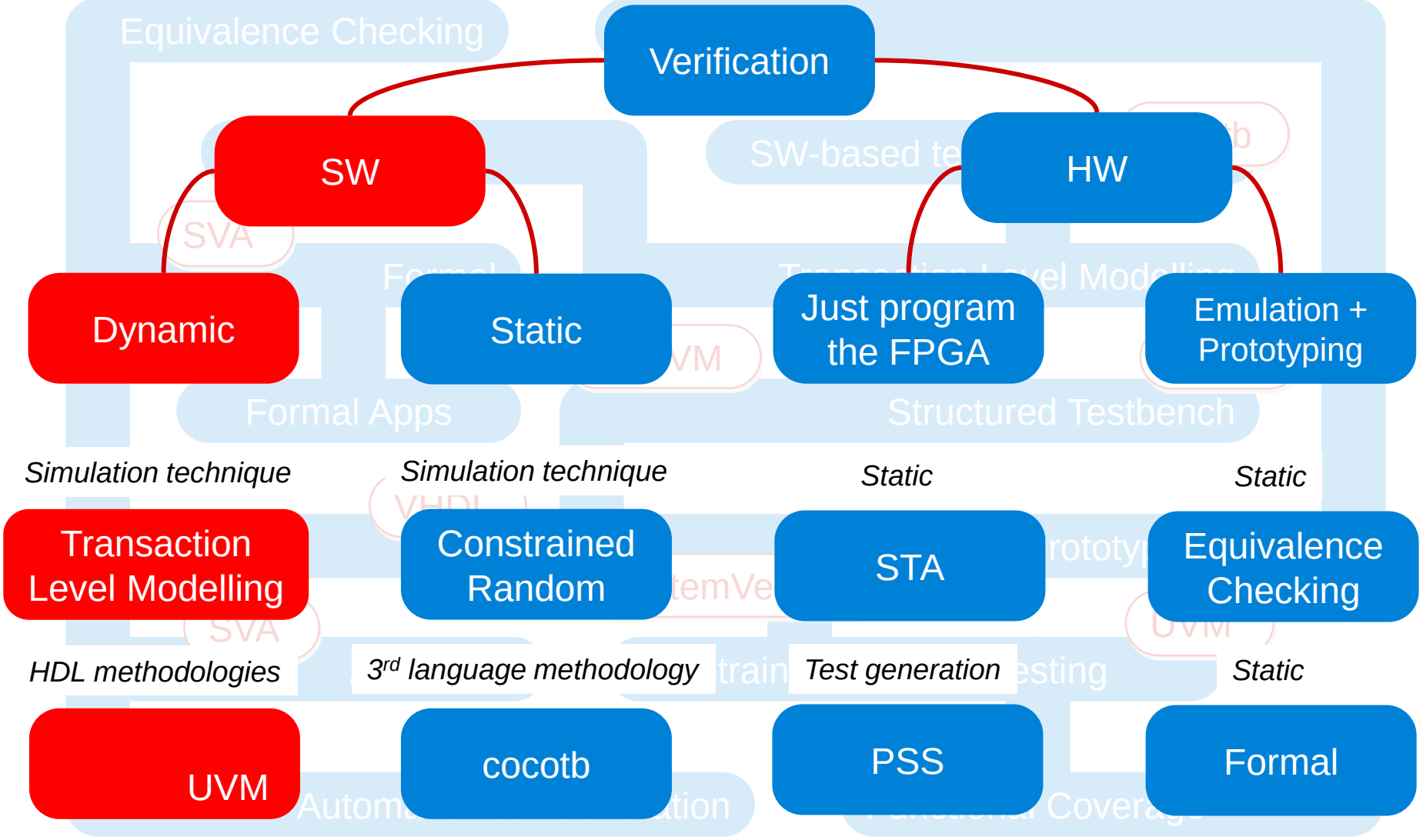
Copyright © 2015-2025 Doulos. All Rights Reserved

UVVM Test

```
...
disable_log_msg(WISHBONE_VVCT, 1, ALL_MESSAGES);
enable_log_msg(WISHBONE_VVCT, 1, ID_BFM);

log(ID_LOG_HDR, "Starting simulation of TB for wb_ram", C_SCOPE);

log("Wait 10 clock period for reset to be turned off");
wait for (10 * C_CLK_PERIOD);

log(ID_LOG_HDR, "Test a write and a read", C_SCOPE);
wishbone_write( WISHBONE_VVCT, 1, X"00000000", X"12345678", "WRITE");
await_completion(WISHBONE_VVCT, 1, 10 * C_CLK_PERIOD);
wait for (C_CLK_PERIOD);

wishbone_check( WISHBONE_VVCT, 1, X"00000000", X"12345678", "CHECK");
await_completion(WISHBONE_VVCT, 1, 10 * C_CLK_PERIOD);

report_alert_counters(FINAL);
log(ID_LOG_HDR, "SIMULATION COMPLETED", C_SCOPE);
...
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



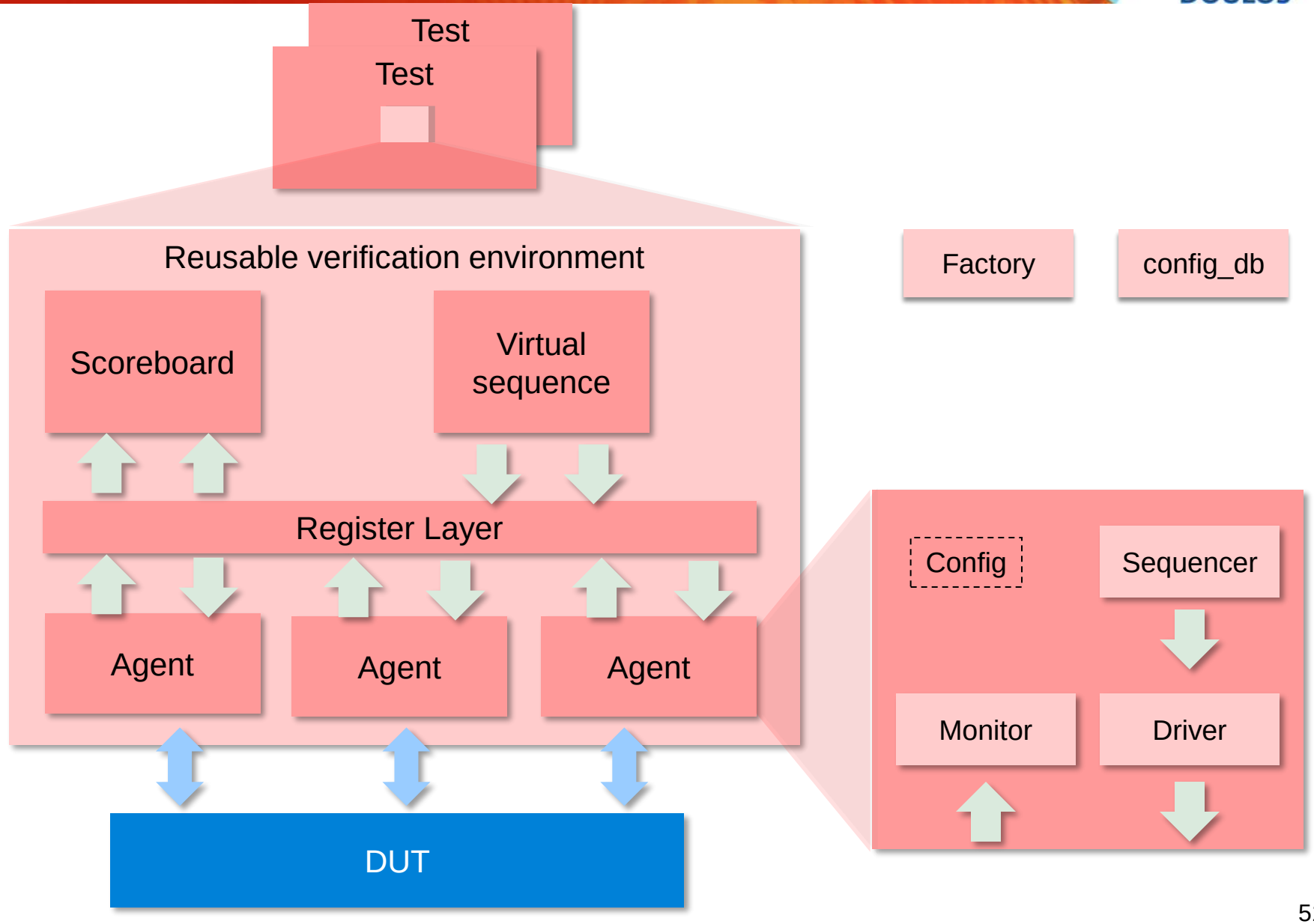
Copyright © 2015-2025 Doulos. All Rights Reserved

What is UVM?

- The Universal Verification Methodology for SystemVerilog
- Supports constrained random, coverage-driven verification
- An open-source (Apache 2.0) base class library
- An Accellera standard and the IEEE Standard 1800.2
- Supported by all major simulator vendors

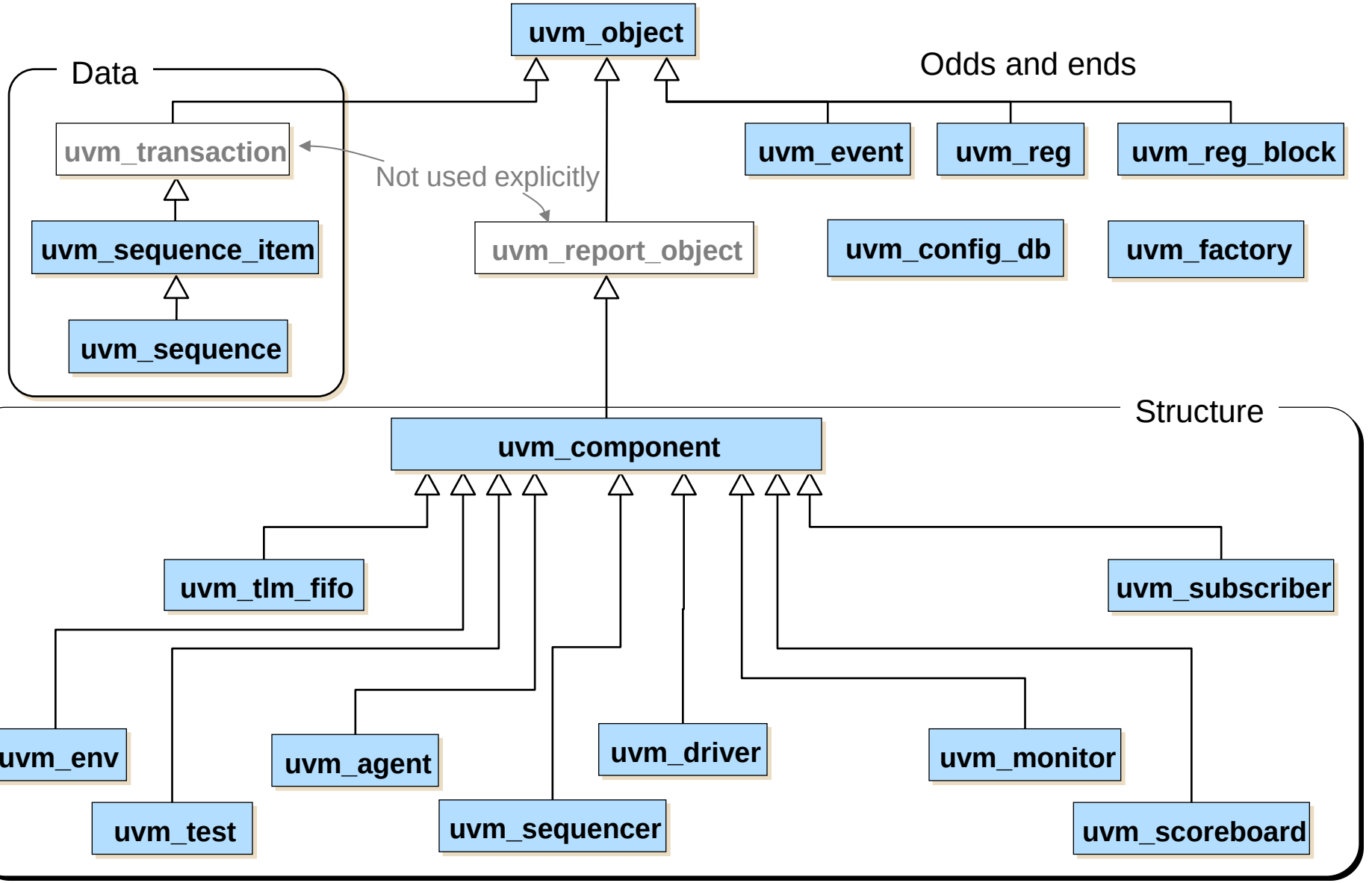
Copyright © 2015-2025 Doulos. All Rights Reserved

The Component Hierarchy



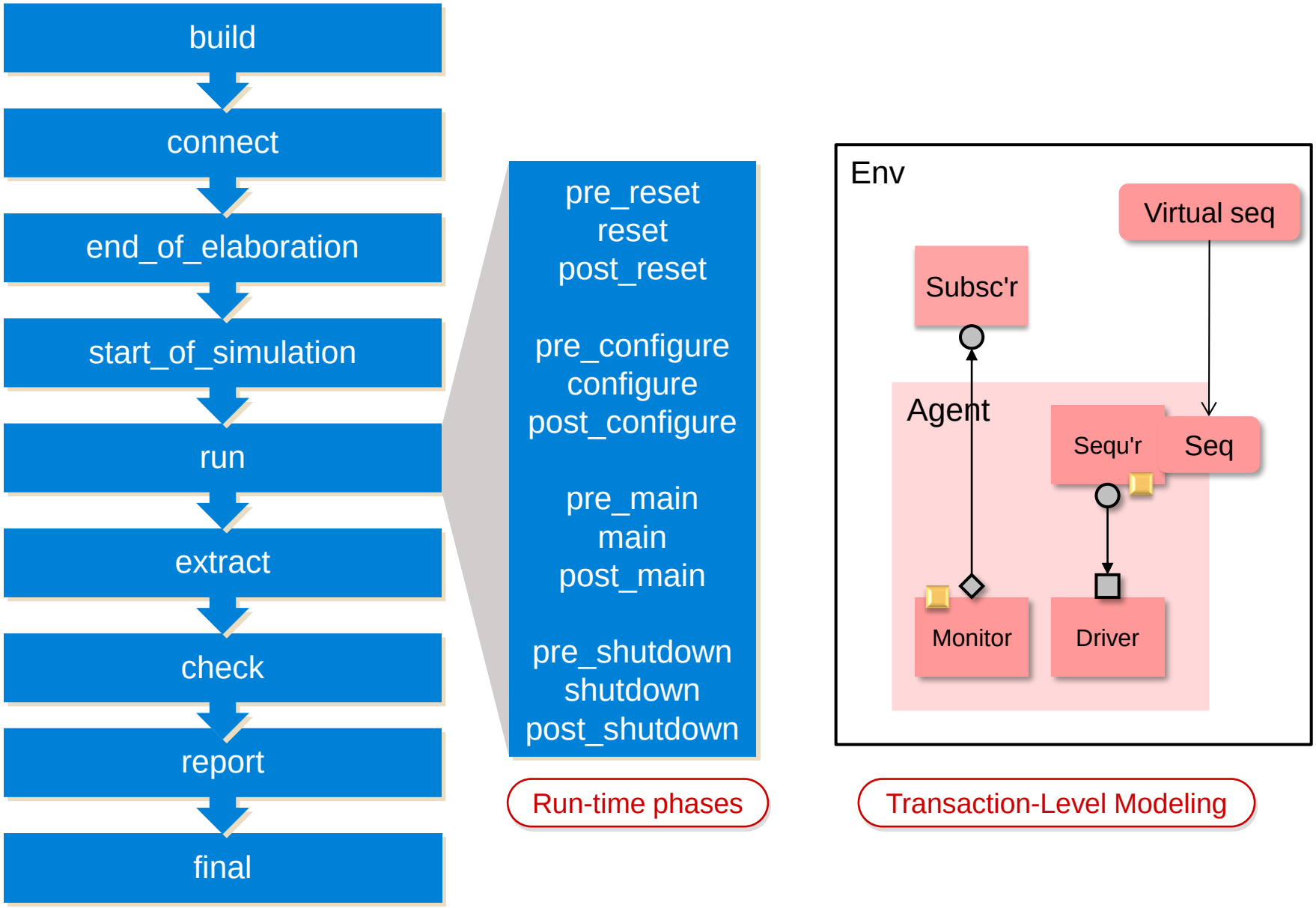
Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Class Hierarchy



Copyright © 2015-2025 Doulos. All Rights Reserved

Simulation Phases



Copyright © 2015-2025 Doulos. All Rights Reserved

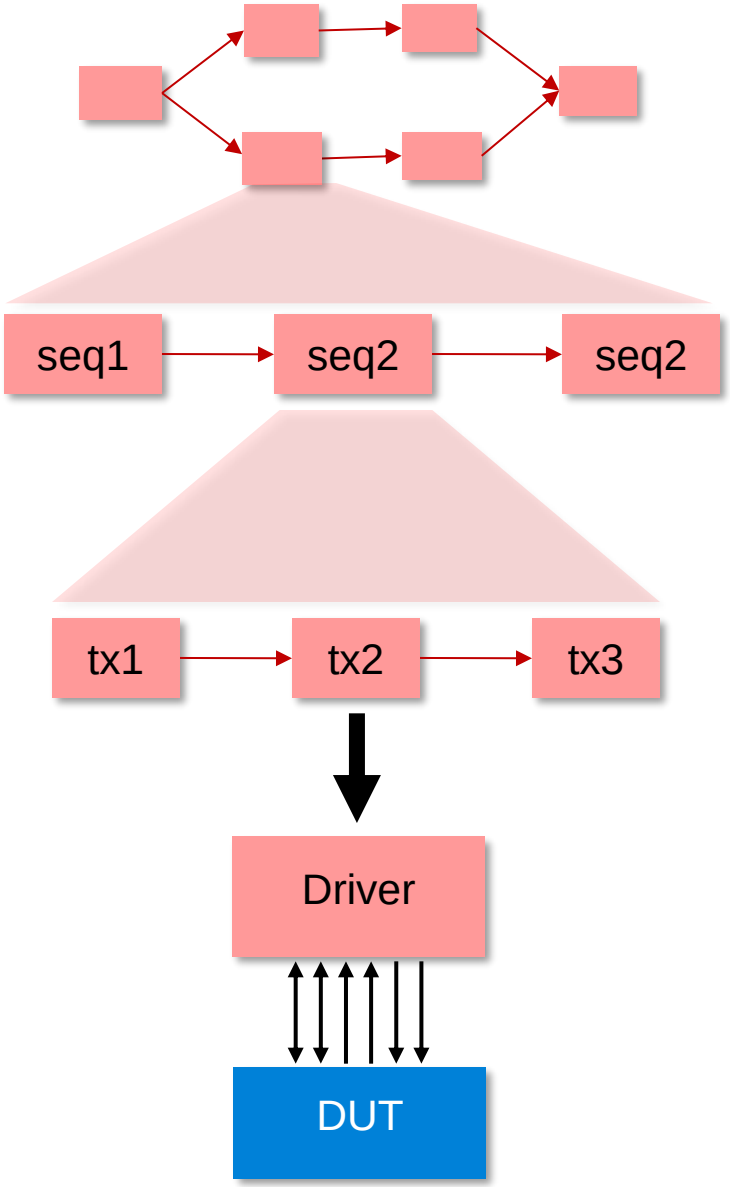
Layered Sequential Stimulus

Envs or tests run virtual sequences

Virtual or layered sequences

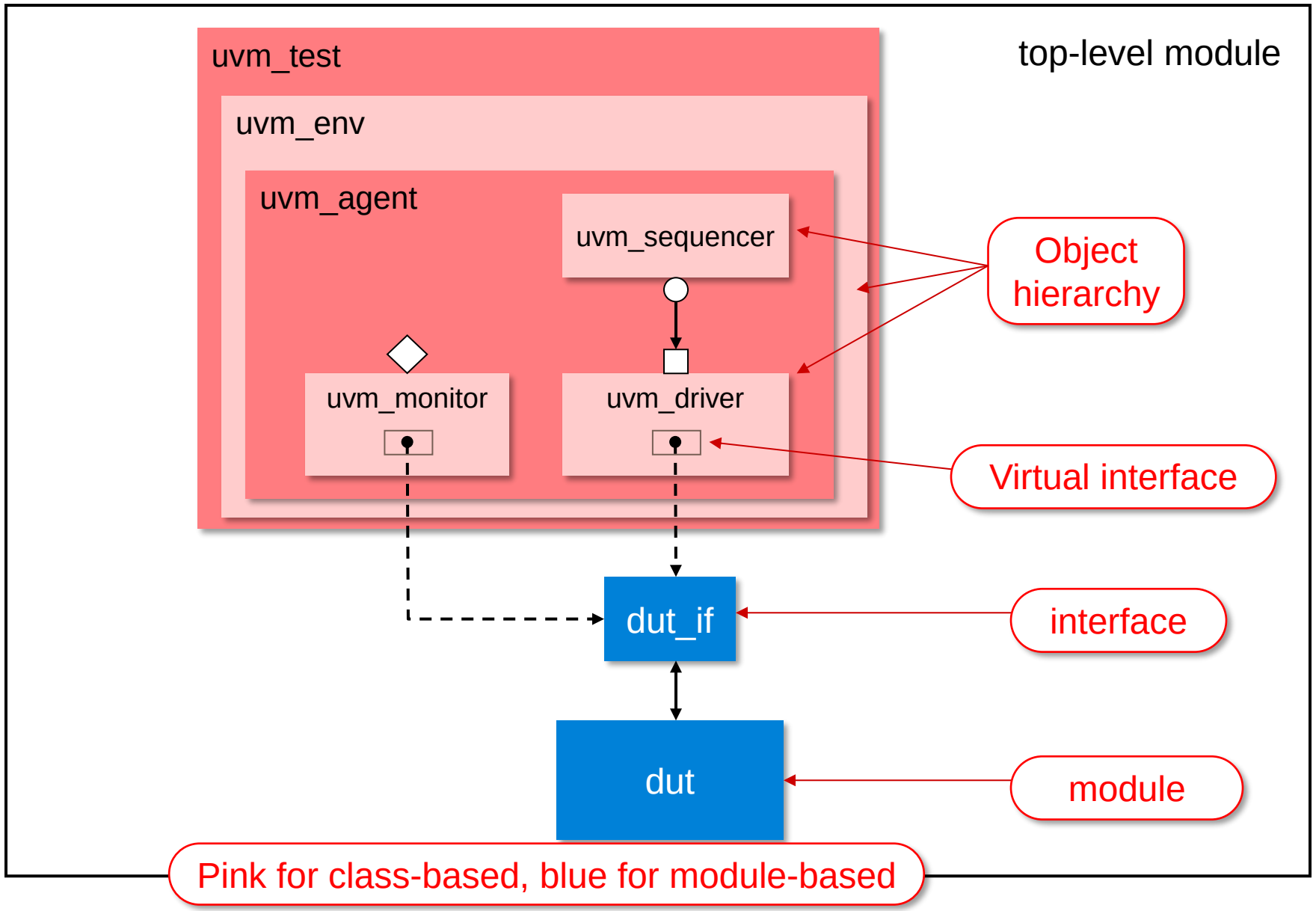
Constrained random
sequence of transactions

Drive transactions into DUT



Copyright © 2015-2025 Doulos. All Rights Reserved

Simple Test Bench Architecture



Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Sequence Item

- aka a transaction

```
class wb_trans extends uvm_sequence_item;

  ...

  rand bit [AWIDTH-1:0] addr;
  rand bit [DWIDTH-1:0] data;
  rand wb_trans_t kind;

endclass
```

Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Sequence

- a UVM sequence is a bunch of transactions
- but it is active – it is a container of code that generates transactions

```
class wb_ram_seq extends uvm_sequence #( wb_trans );

  ...

  task body();
    repeat(10)
      `uvm_do_with( req, { kind inside {WRITE,READ};
                           addr < MEMORY_SIZE;} )
  endtask

endclass
```

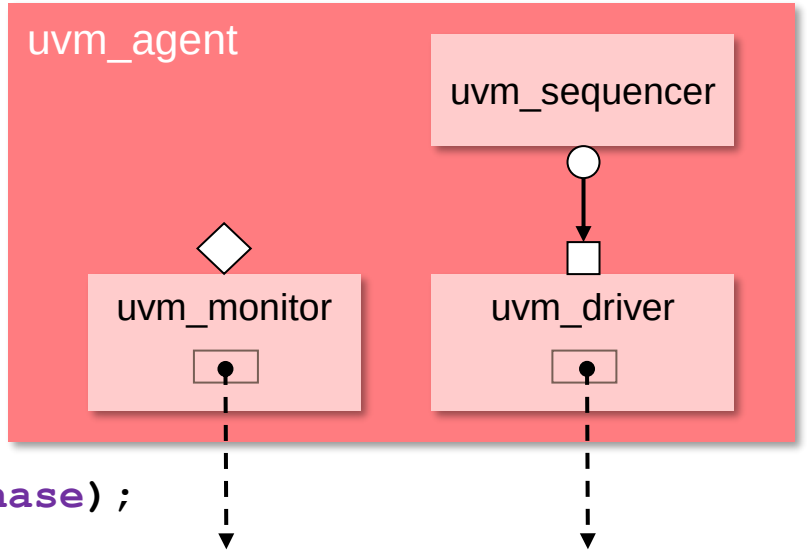
Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Driver

```
class wb_driver extends uvm_driver #( wb_trans );
  ...

  virtual task run_phase(uvm_phase phase);
    ...
    forever begin
      seq_item_port.get_next_item( req );
      case (req.kind )
        READ  : wb_read (req.addr, req.data );
        WRITE : wb_write(req.addr, req.data );
      endcase
      seq_item_port.item_done( req );
    end
  endtask

  ...
endclass
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Agent




```
class wb_agent extends uvm_agent;
  ...
  wb_monitor   m_wb_mon;
  wb_driver    m_wb_drv;
  wb_sequencer m_wb_seqr;

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    m_wb_drv  = wb_driver   ::type_id::create( "m_wb_drv",  this );
    m_wb_seqr = wb_sequencer::type_id::create( "m_wb_seqr", this );
    m_wb_mon  = wb_monitor  ::type_id::create( "m_wb_mon",  this );
  endfunction

  function void connect_phase(uvm_phase phase);
    m_wb_drv.seq_item_port.connect( m_wb_seqr.seq_item_export );
  endfunction

endclass
```

Copyright © 2015-2025 Doulos. All Rights Reserved

UVM “env”

```
class wb_ram_env extends uvm_env;

  ...

  wb_agent                    m_wb_agent;
  wb_ram_sb                   m_wb_ram_sb;

  function void build_phase(uvm_phase phase);
    m_wb_agent  = wb_agent ::type_id::create( "m_wb_agent",  this );
    m_wb_ram_sb = wb_ram_sb::type_id::create( "m_wb_ram_sb", this );
  endfunction

  function void connect_phase(uvm_phase phase);
    m_wb_agent.m_wb_mon.sb_ap.connect(
                              m_wb_ram_sb.wb_ap_fifo.analysis_export );
  endfunction

endclass
```

Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Test

```
class wb_ram_test extends uvm_test;
  ...

  wb_ram_env  m_env;

  function void build_phase(uvm_phase phase);
    m_env = wb_ram_env::type_id::create("m_env", this);
  endfunction

  task run_phase(uvm_phase phase);
    wb_ram_seq seq;
    phase.raise_objection(this, "Starting Test");
    seq = wb_ram_seq::type_id::create("seq");
    void'(seq.randomize());
    seq.start(m_env.m_wb_agent.m_wb_seqr);
    phase.drop_objection(this, "Test Finished");
  endtask

endclass
```

Copyright © 2015-2025 Doulos. All Rights Reserved

UVM Top-level Module

```
`include "uvm_macros.svh"

module wb_ram_tb;
  import uvm_pkg::*; import uvm_ram_test_pkg::*;
  ...
  wb_ram
    #(.depth (MEMORY_SIZE))
  dut
    (.wb_clk_i (wb_clk), .wb_rst_i (wb_rst),
     .wb_adr_i (wb_adr[$clog2(MEMORY_SIZE)-1:0]),
     ...
     .wb_ack_o (wb_ack));

 initial begin
    ...
    run_test();
  end
...
```

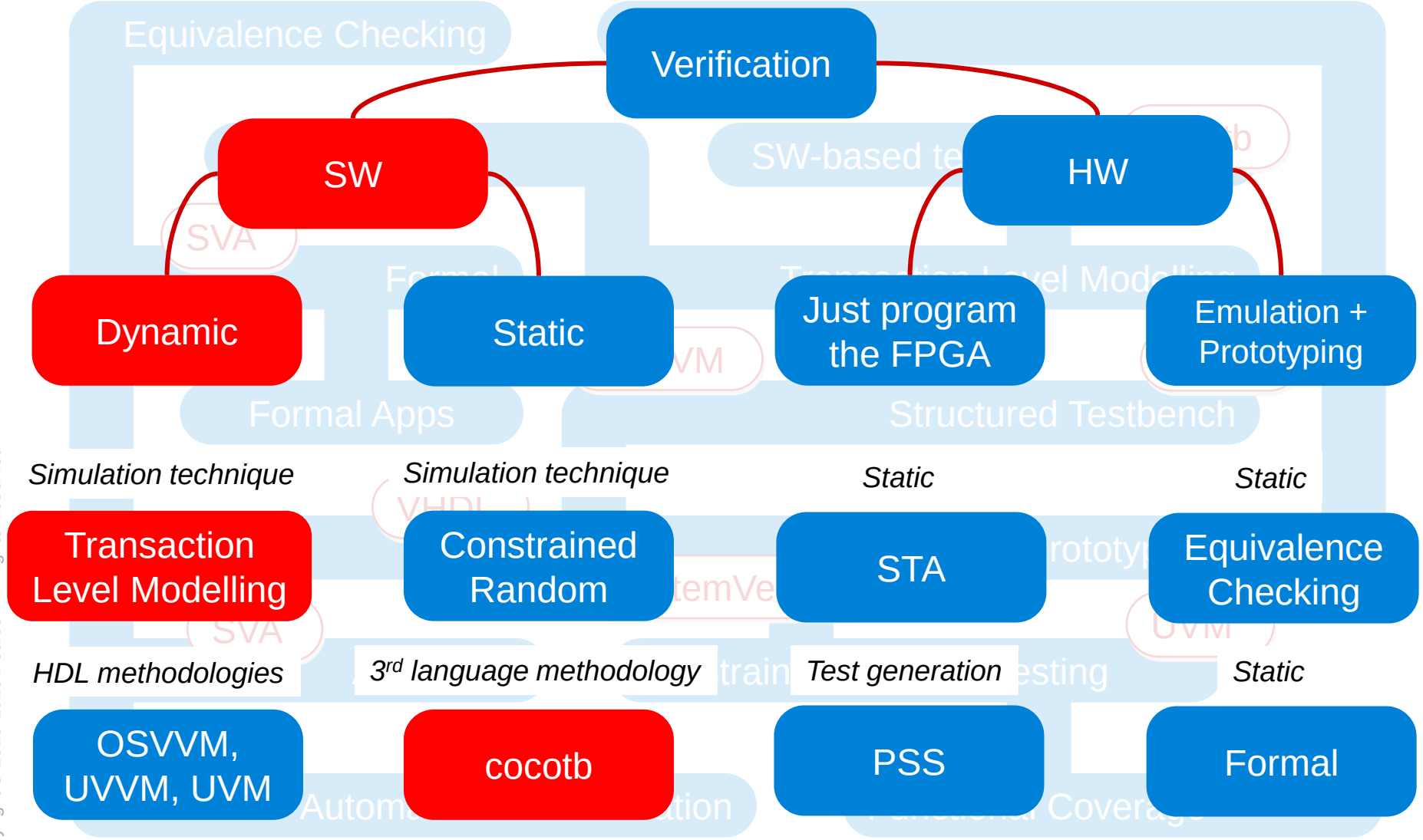
DUT

Test name can be given on the command line

```
$> <your_simulator> +UVM_TESTNAME=<test name>
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



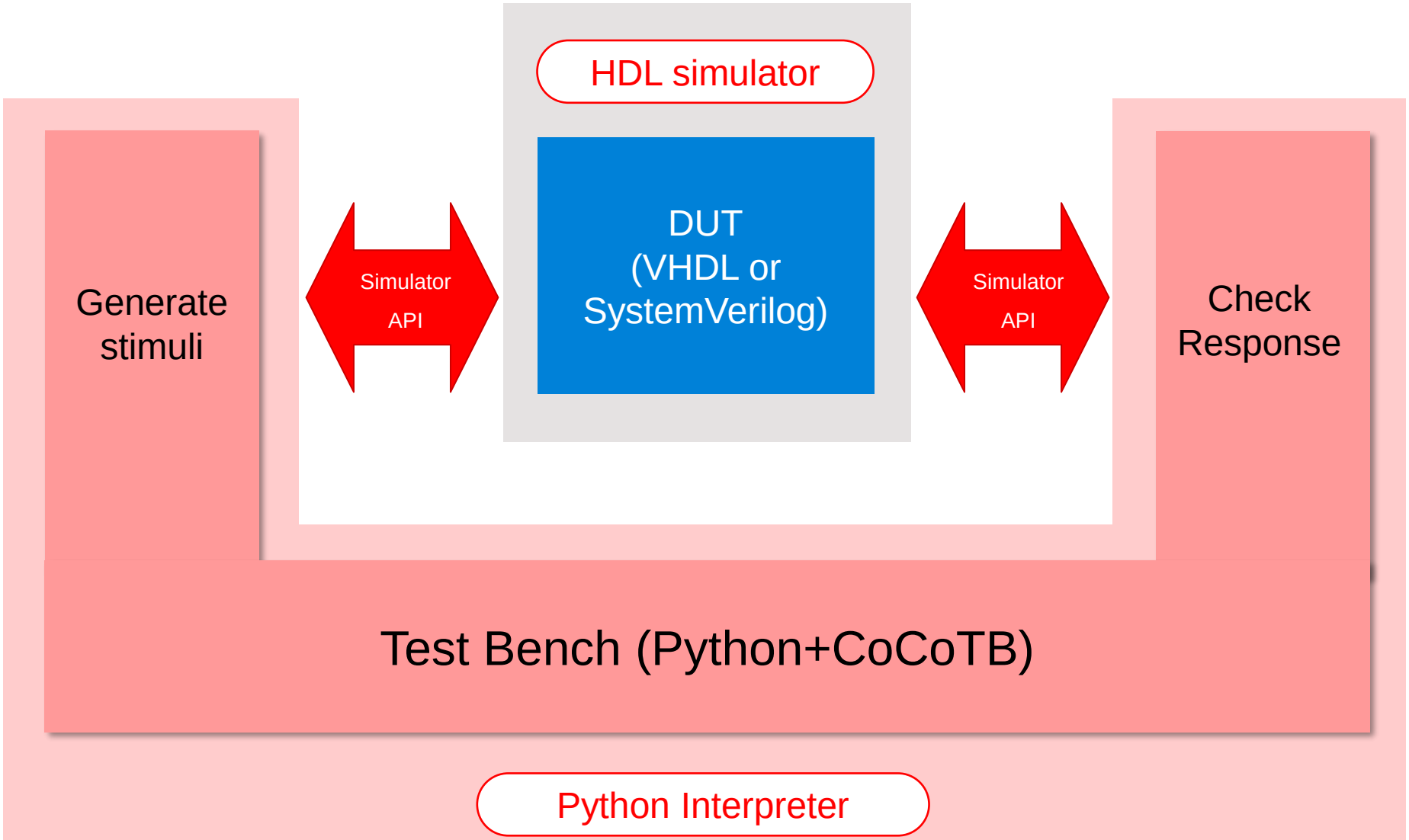
Copyright © 2015-2025 Doulos. All Rights Reserved

About CoCoTB

- *Co*-routine-based *Co*-simulation *T*est *B*ench environment
- Created by Philipp Wagner and others
- Extends Python to support hardware verification
  - Enables reading and writing of HDL design elements
  - Handles concurrency and synchronization elegantly
  - Makes it easier to create verification components
- Open-source (BSD licensed)
- Works with a growing number of commercial and free simulators
- www.cocotb.org

Copyright © 2015-2025 Doulos. All Rights Reserved

CoCoTB Co-Simulation



Copyright © 2015-2025 Doulos. All Rights Reserved

cocotb – BFM function (write)


```
@cocotb.coroutine
def write_ram(dut, address, value):
    """This coroutine performs a write of the RAM"""
    yield RisingEdge(dut.wb_clk_i)
    dut.wb_cyc_i.value = 0
    dut.wb_stb_i.value = 0
    dut.wb_adr_i.value = address
    dut.wb_dat_i.value = value
    dut.wb_sel_i.value = 15
    dut.wb_we_i.value  = 1
    yield RisingEdge(dut.wb_clk_i)
    dut.wb_cyc_i.value = 1
    dut.wb_stb_i.value = 1
    yield RisingEdge(dut.wb_clk_i)
```

Copyright © 2015-2025 Doulos. All Rights Reserved

cocotb – BFM function (read)


```
@cocotb.coroutine
def read_ram(dut, address):
    """This coroutine performs a read of the RAM and returns
a value"""
    yield RisingEdge(dut.wb_clk_i)
    dut.wb_cyc_i.value = 0
    dut.wb_stb_i.value = 0
    dut.wb_adr_i.value = address
    dut.wb_sel_i.value = 15
    dut.wb_we_i.value  = 0
    yield RisingEdge(dut.wb_clk_i)
    dut.wb_cyc_i.value = 1
    dut.wb_stb_i.value = 1
    yield RisingEdge(dut.wb_clk_i)
    yield ReadOnly()
    return int(dut.wb_dat_o.value)
```

Copyright © 2015-2025 Doulos. All Rights Reserved

cocotb – test

```
@cocotb.test()
def test_ram(dut):
  RAM = {}
  width = 32
  depth = 256

  cocotb.fork(Clock(dut.wb_clk_i, 3200).start())

  dut._log.info("Writing in random values")
  for i in range(depth):
    RAM[i] = int(random.getrandbits(width))
    yield write_ram(dut, i*4, RAM[i])

  dut._log.info("Reading back values and checking")
  for i in range(depth):
    value = yield read_ram(dut, i*4)
    if value != RAM[i]:
      dut._log.error("RAM[%d] expected %d but got %d" % (i, RAM[i],
                                                         dut.wb_dat_o.value.value))
      raise TestFailure("RAM contents incorrect")
    dut._log.info("RAM contents OK")
```

Copyright © 2015-2025 Doulos. All Rights Reserved

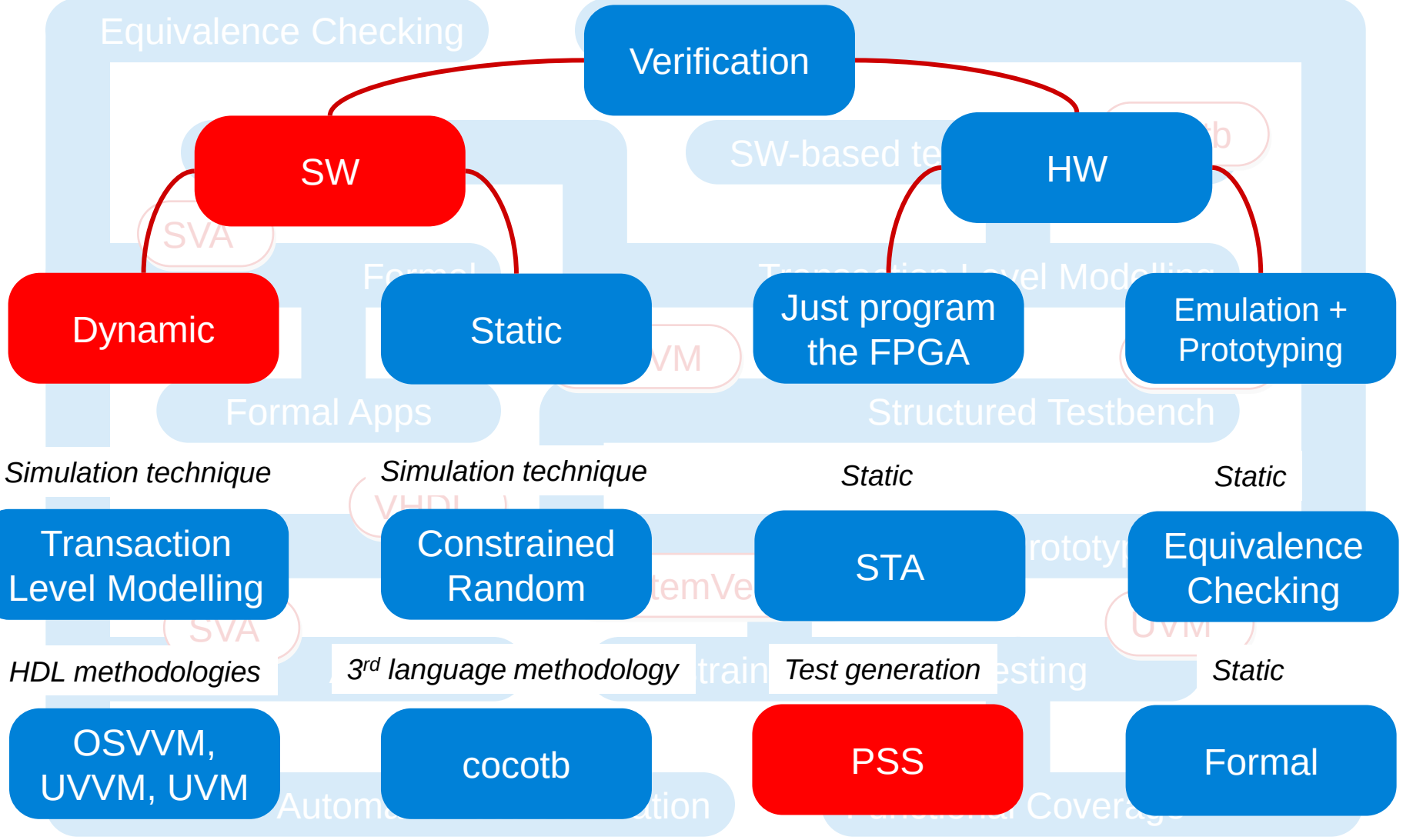
cocotb – “test runner”

```
def test_runner():

  import os
  from cocotb.runner import get_runner

  runner = get_runner(os.getenv("SIM"))
  runner.build(
    verilog_sources=["design.sv"],
    hdl_toplevel="wb_ram",
    always=True,
  )

  runner.test(hdl_toplevel="wb_ram", test_module="test bench,")

if __name__ == "__main__":
    test_runner()
```

https://www.edaplayground.com/x/XeeV

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

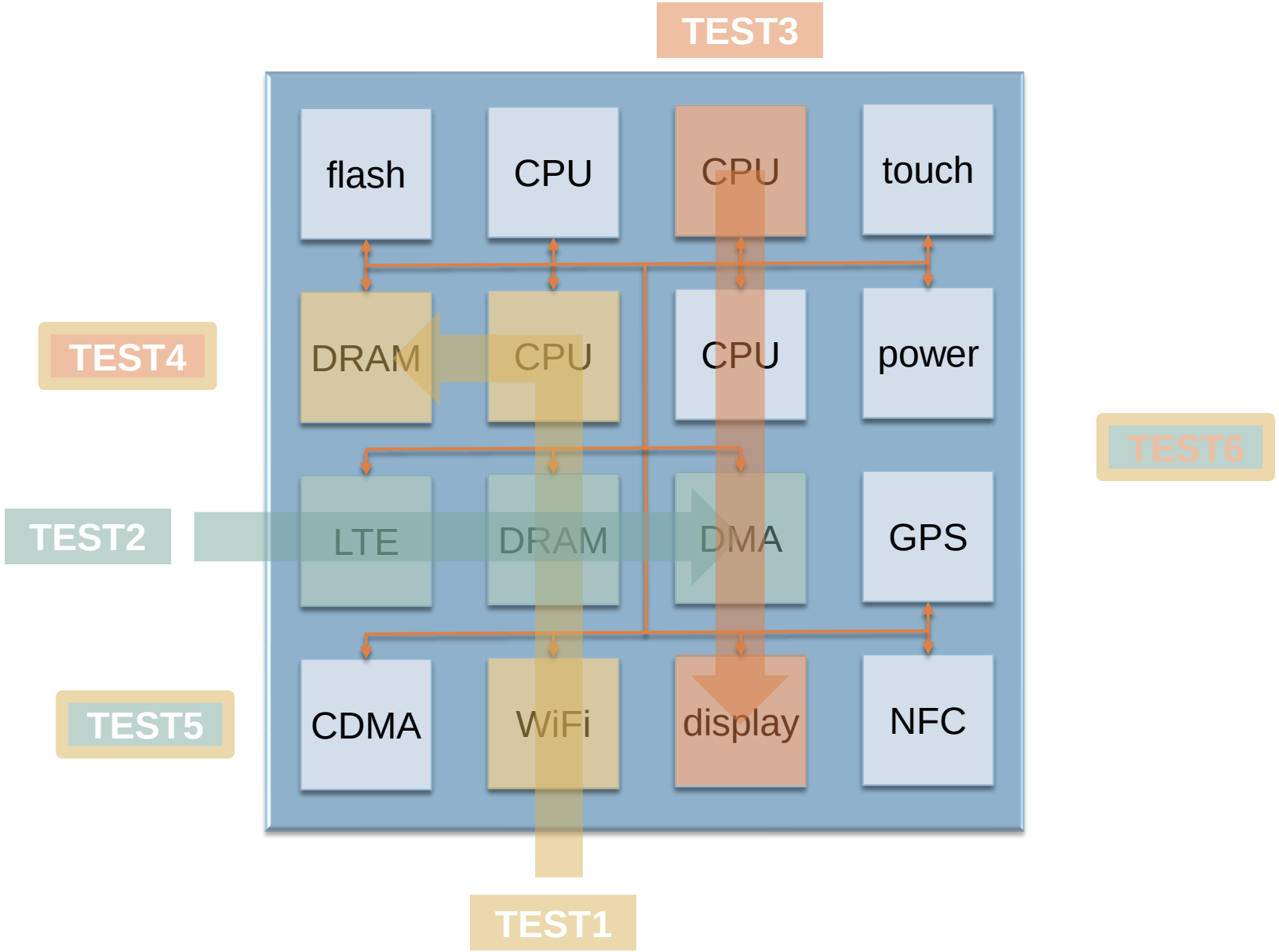
What is PSS?

- **P**ortable Test and **S**timulus **S**tandard
- A language
- Accellera Standard:

Portable Test and Stimulus Standard

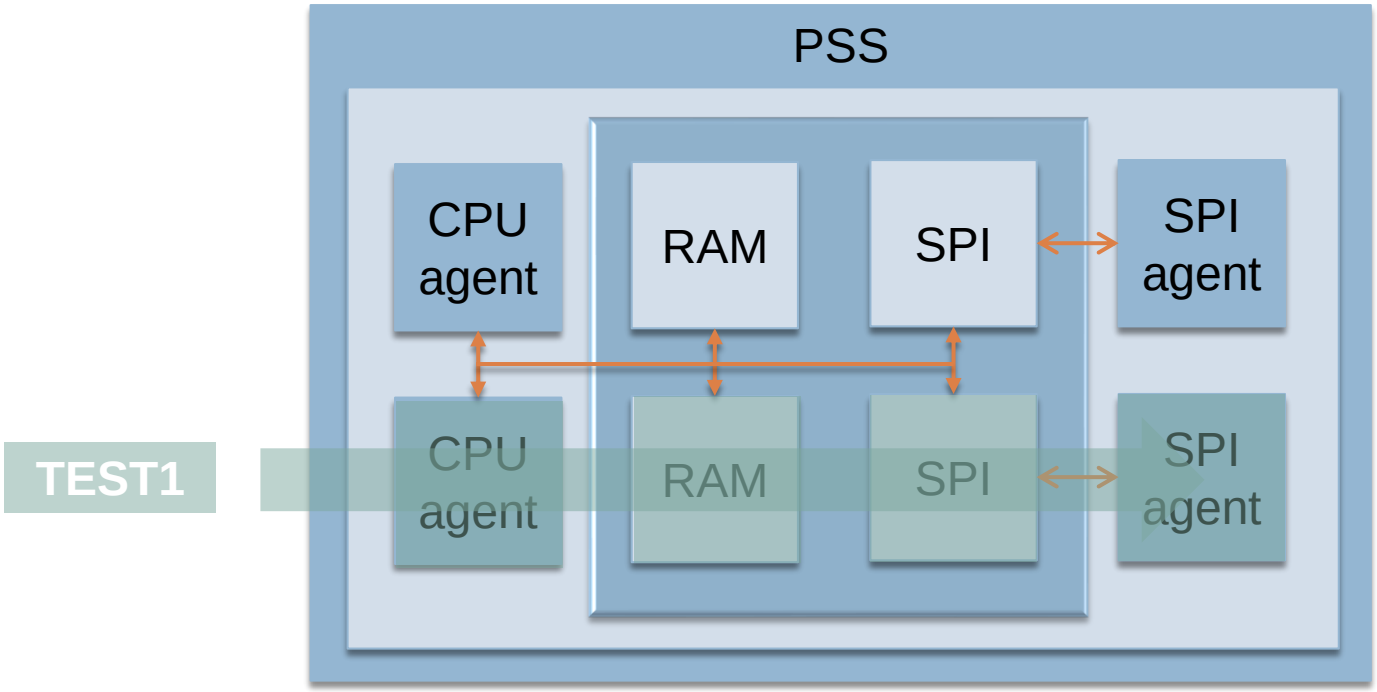
Copyright © 2015-2025 Doulos. All Rights Reserved

The next step beyond constrained random



Copyright © 2015-2025 Doulos. All Rights Reserved

PSS+UVM Test Bench



Copyright © 2015-2025 Doulos. All Rights Reserved

Resources

```
package rsrc_pkg {

  import wb_params_pkg::*;

  resource cpu_r {
    rand int in [0..NCPUS-1] cpu_id;
    constraint {cpu_id == instance_id;}
    }

  resource mem_r {
    rand int in [0..NMEMS-1] mem_id;
    constraint {mem_id == instance_id;}
    }

  resource spi_r {
    rand int in [0..NSPIS-1] spi_id;
    constraint {spi_id == instance_id;}
    }

}
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Resources

```
component pss_top {
  ...
  pool [NCPUS] cpu_r cpu_rp;
  bind cpu_rp *;

  pool [NMEMS] mem_r mem_rp;
  bind mem_rp *;

  pool [NSPIS] spi_r spi_rp;
  bind spi_rp *;

  action ref_model {
    spi_c::a_mem_to_spi  mem_to_spi_0, mem_to_spi_1;

    activity {
      parallel {
        mem_to_spi_0;
        mem_to_spi_1;
      }
    }
  ...
```

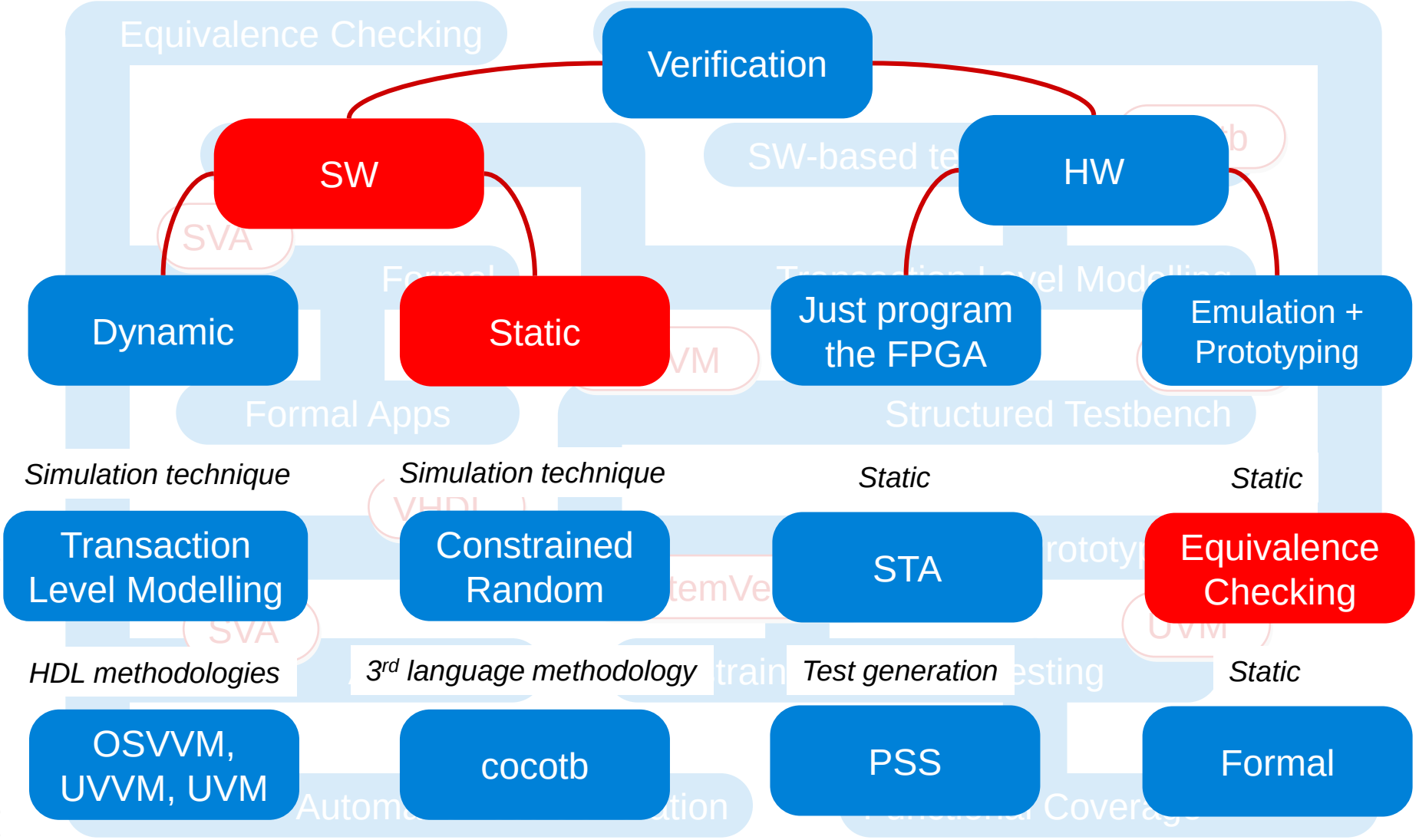
Copyright © 2015-2025 Doulos. All Rights Reserved

Resources


```
component spi_c {

state spi_config_s {
    rand spi_state_e spi_state;
    constraint initial -> spi_state == NOT_SETUP;
    rand int spi_id;
  }

  action a_setup_spi {

    lock spi_r spi;
    lock cpu_r cpu;

    output spi_config_s spi_config;
    constraint { spi_config.spi_state == SETUP; }
    constraint { spi_config.spi_id == spi.spi_id; }

    exec body {
      f_spi_setup( cpu_to_spi, spi.spi_id );
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques

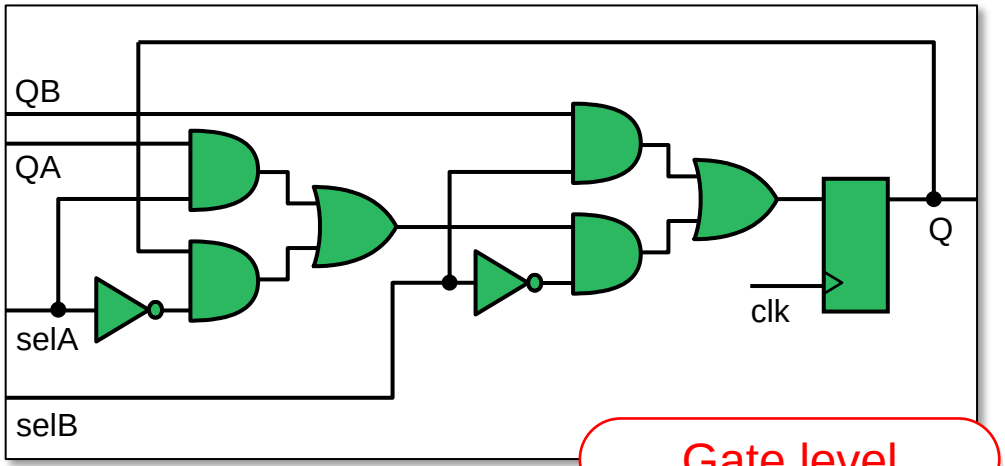


Copyright © 2015-2025 Doulos. All Rights Reserved

Logic Equivalency Checking

RTL

```
module selAB (
  input  logic clk,
  input  logic QA, selA, QB, selB,
  output logic Q
);

  always @(posedge clk)
    begin
      if (selA) Q <= QA;
      if (selB) Q <= QB;
    end

endmodule
```



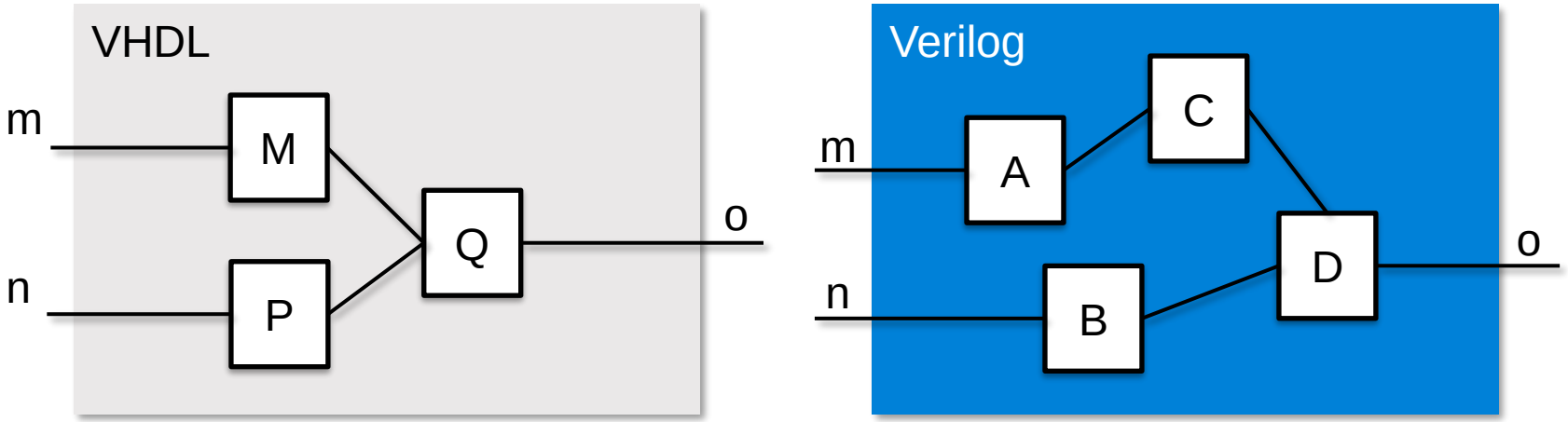
Gate level

Are they functionally the same?

- RTL versus gate-level netlist
- Netlist versus netlist
- Only works with recognizable equivalency points (signal names)
- Retiming may break LEC (needs same number of registers)

Copyright © 2015-2025 Doulos. All Rights Reserved

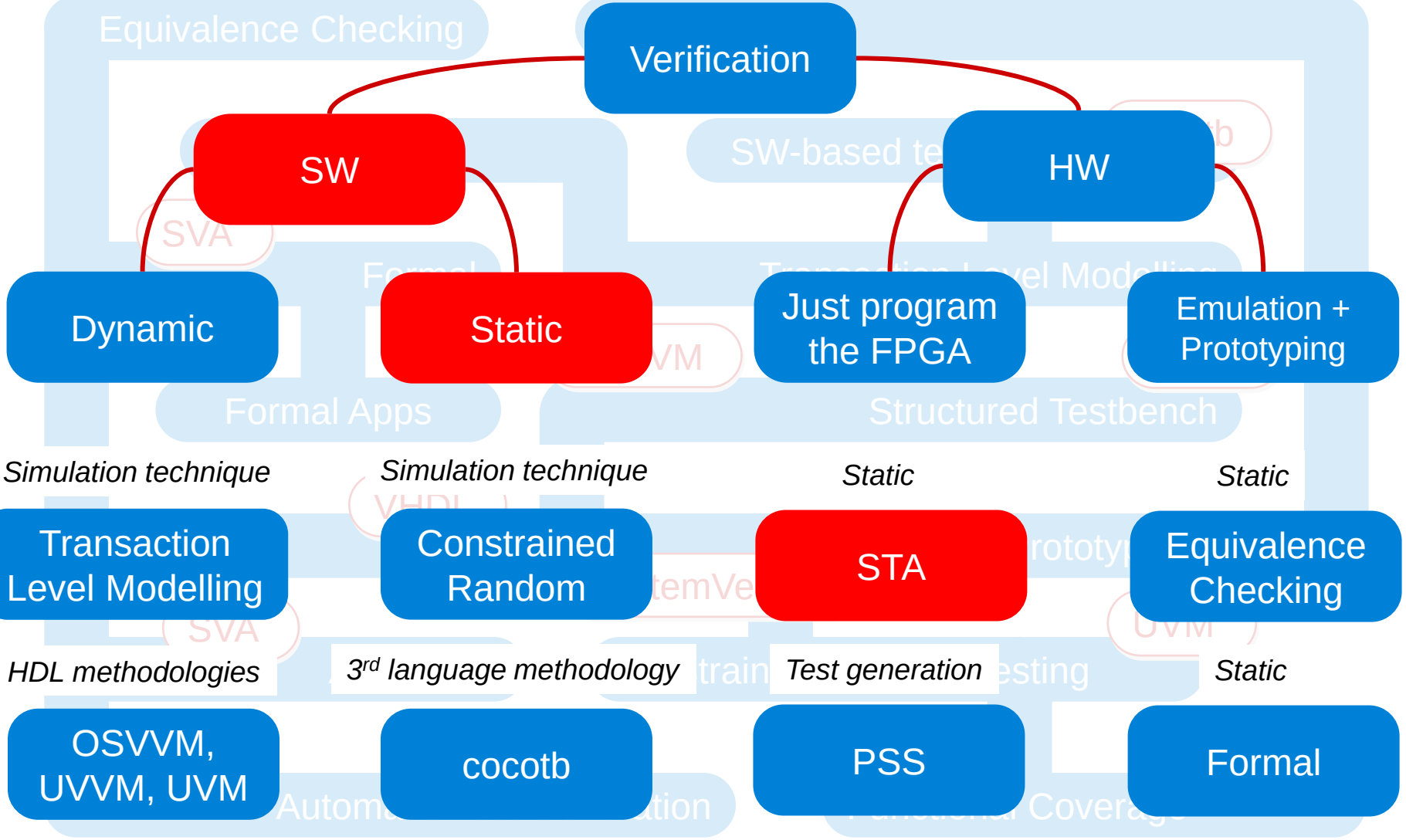
Sequential Equivalency Checking



- Dynamic, not static like LEC – advances the clock
- Shows equivalency between different implementations
- Equivalency at the port-level
- RTL <-> RTL, RTL <-> HLS (SystemC/C/C++)

Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques

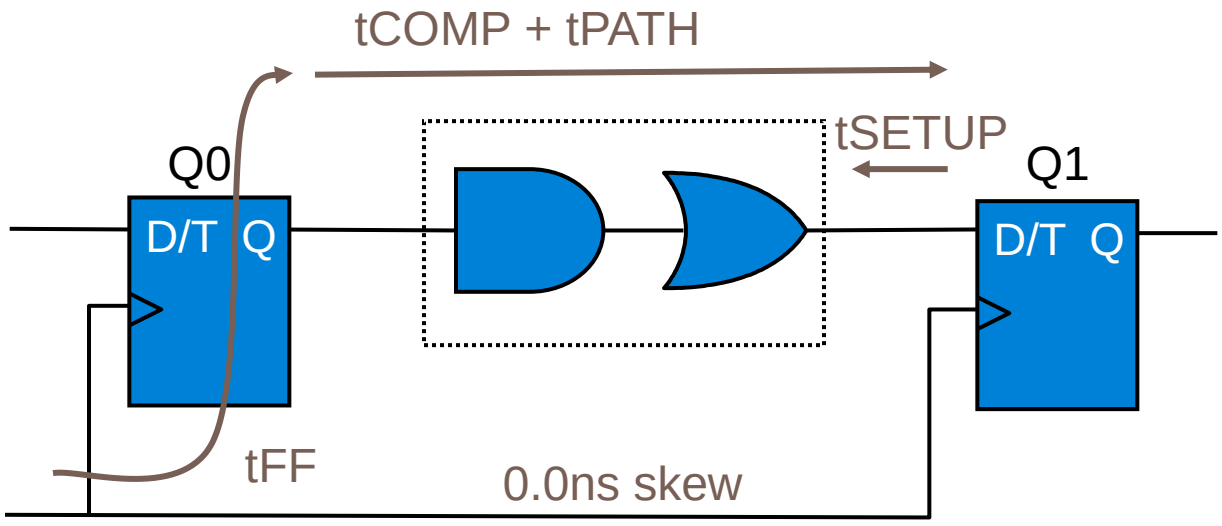


Copyright © 2015-2025 Doulos. All Rights Reserved

Static Timing Analysis Example

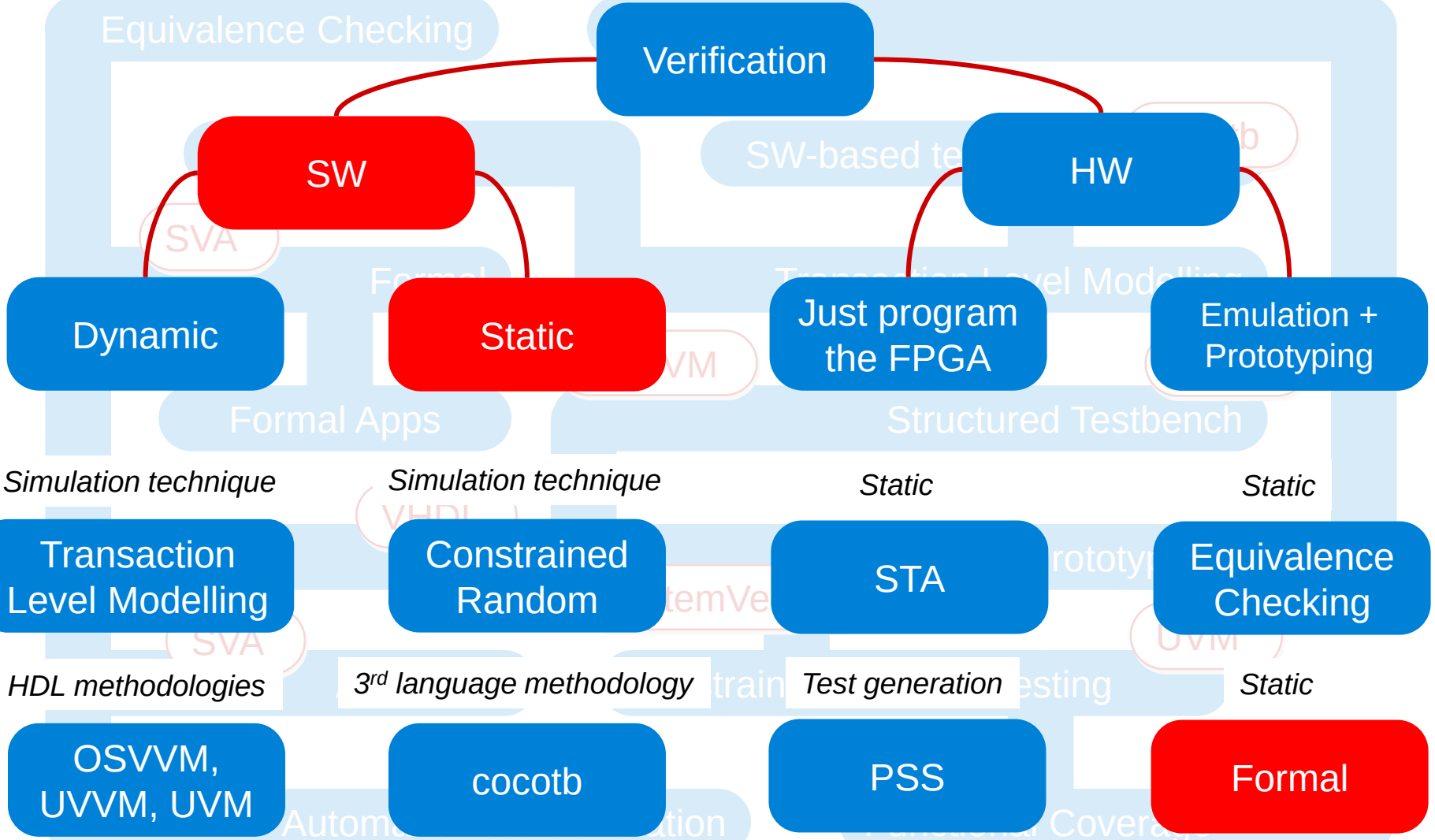
```
Source clock net: Q0.C
Worst case clock delay from origin Clock is  0.7 ns
Destination clock net: Q1.C
Worst case clock delay from origin Clock is  0.7 ns
Source and destination clock skew:  0.0 ns
From: Q0.Q      tFF                              :  1.0 ns (1.0 ns)
  To: Q1.T      tCOMP + tPATH + tSETUP           :  4.7 ns (5.7 ns)
Target Clock period: 10 ns
```

Slack time: 4.3 ns



Copyright © 2015-2025 Doulos. All Rights Reserved

Verification Techniques



Copyright © 2015-2025 Doulos. All Rights Reserved

Formal

```
bit [31:0] d, dn;
bit [5:0]  a, an;
bit [ 1:0] I, in;

default clocking cb @(posedge wb_clk_i); endclocking

asm_constant_data : assume property ( ##1 $stable(d) );
asm_another_addr  : assume property ( a != an );
asm_constant_addr : assume property ( ##1 $stable(a) );
asm_constant_index: assume property ( ##1 $stable(i) );
```

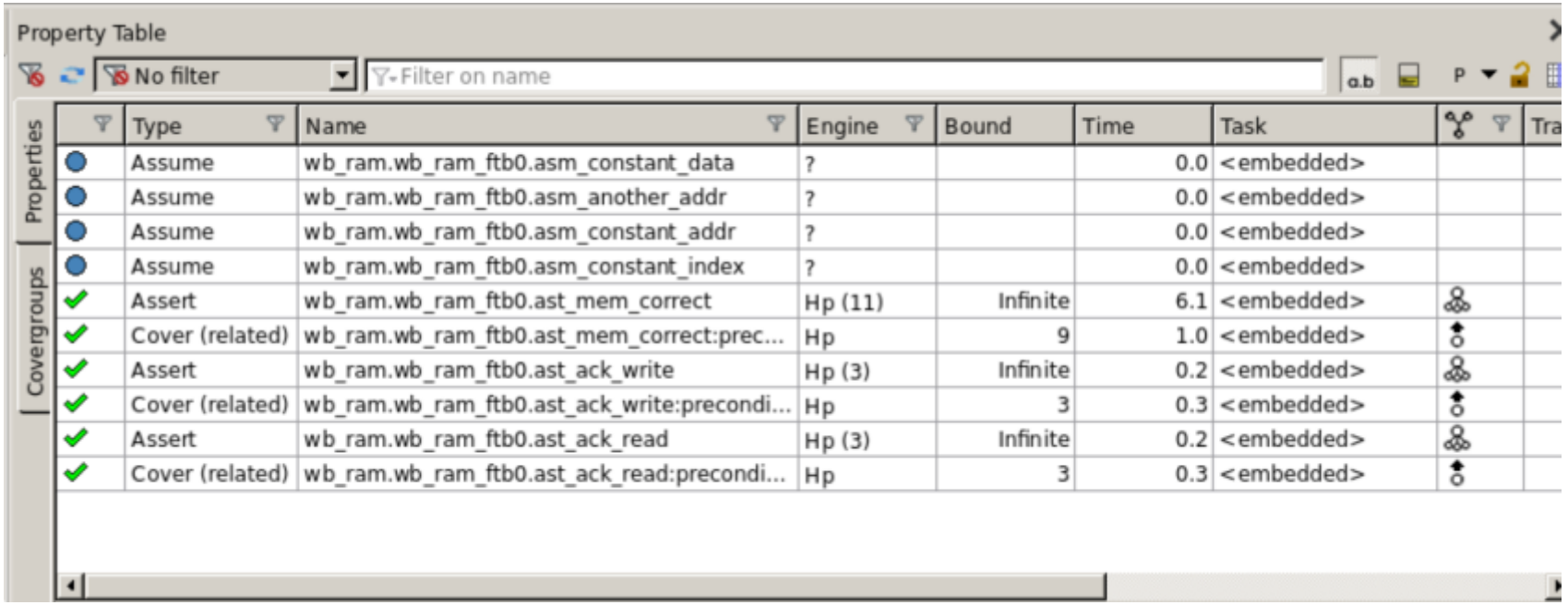
Copyright © 2015-2025 Doulos. All Rights Reserved

Formal

```
sequence write (addr, data, sel);
    ( wb_adr_i==addr && wb_dat_i==data && wb_sel_i==2**sel &&  wb_we_i )
  throughout
    ( (!wb_cyc_i && !wb_stb_i) ##1 (wb_cyc_i && wb_stb_i)[*2] );
endsequence

sequence read (addr, sel);
    ( wb_adr_i==addr &&                    wb_sel_i==2**sel && !wb_we_i )
  throughout
    ( (!wb_cyc_i && !wb_stb_i) ##1 (wb_cyc_i && wb_stb_i)[*2] );
endsequence

ast_mem_correct: assert property (
  write(a*4, d, i) ##1
 (write(an*4, dn, in) or read(an*4, in))[*1:2] ##1 read(a*4, i) |->
  wb_dat_o[8*i+:8] == d[8*i+:8]
);

ast_ack_write: assert property ( write(a*4, d, i).triggered() |-> wb_ack_o );

ast_ack_read : assert property (  read(a*4, d, i).triggered() |-> wb_ack_o );
```

Copyright © 2015-2025 Doulos. All Rights Reserved

Formal



Type	Name	Engine	Bound	Time	Task	Tra
Assume	wb_ram.wb_ram_ftb0.asm_constant_data	?		0.0	<embedded>	
Assume	wb_ram.wb_ram_ftb0.asm_another_addr	?		0.0	<embedded>	
Assume	wb_ram.wb_ram_ftb0.asm_constant_addr	?		0.0	<embedded>	
Assume	wb_ram.wb_ram_ftb0.asm_constant_index	?		0.0	<embedded>	
Assert	wb_ram.wb_ram_ftb0.ast_mem_correct	Hp (11)	Infinite	6.1	<embedded>	
Cover (related)	wb_ram.wb_ram_ftb0.ast_mem_correct:prec...	Hp	9	1.0	<embedded>	
Assert	wb_ram.wb_ram_ftb0.ast_ack_write	Hp (3)	Infinite	0.2	<embedded>	
Cover (related)	wb_ram.wb_ram_ftb0.ast_ack_write:precondi...	Hp	3	0.3	<embedded>	
Assert	wb_ram.wb_ram_ftb0.ast_ack_read	Hp (3)	Infinite	0.2	<embedded>	
Cover (related)	wb_ram.wb_ram_ftb0.ast_ack_read:precondi...	Hp	3	0.3	<embedded>	

Copyright © 2015-2025 Doulos. All Rights Reserved

Common Formal Apps

- Names differ between tool vendors
- Similarly named apps might not do exactly the same thing
- There exist other apps not shown here

Formal property checking

Automatic property checking

Coverage analysis

X-Propagation checking

Connectivity checking

Control/status register checking

Sequential equivalence checking

Security path checking

Copyright © 2015-2025 Doulos. All Rights Reserved

SoC Design & Verification	» SystemVerilog » UVM » Formal » SystemC » TLM-2.0
FPGA & Hardware Design	» VHDL » Verilog » SystemVerilog » Tcl » AMD » Intel FPGA
Embedded Software & Arm	» Emb C/C++ » Emb Linux » Yocto » RTOS » Security » Android » Arm » Rust » Zephyr
Python, AI & Machine Learning	» Python » Edge AI » Deep Learning

Copy